CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO

Ruan Kenned Martins Alves

ANÁLISE COMPARATIVA DE MÉTODOS DE OTIMIZAÇÃO DAS BIBLIOTECAS SCIPY E PYMOO APLICADOS A UM PROBLEMA DE OTIMIZAÇÃO GEOMÉTRICA UTILIZANDO A INTERFACE PYMAPDL

Timóteo

2024

Ruan Kenned Martins Alves

ANÁLISE COMPARATIVA DE MÉTODOS DE OTIMIZAÇÃO DAS BIBLIOTECAS SCIPY E PYMOO APLICADOS A UM PROBLEMA DE OTIMIZAÇÃO GEOMÉTRICA UTILIZANDO A INTERFACE PYMAPDL

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Prof. Thiago de Sousa Goveia

Timóteo

2024

Ruan Kenned Martins Alves

ANÁLISE COMPARATIVA DE MÉTODOS DE OTIMIZAÇÃO DAS BIBLIOTECAS SCIPY E PYMOO APLICADOS A UM PROBLEMA DE OTIMIZAÇÃO GEOMÉTRICA UTILIZANDO A INTERFACE PYMAPDL

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação do Centro Federal de Educação Tecnológica de Minas Gerais, campus Timóteo, como requisito parcial para obtenção do título de Engenheiro de Computação.

Trabalho aprovado. Timóteo, 12 de Setembro de 2024:

gov.br
Documento assinado digitalmente
THIAGO DE SOUSA GOVEIA
Data: 18/09/2024 23:57:16-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Thiago de Sousa Goveia
Orientador

gov.br
Documento assinado digitalmente
MAURILIO ALVES MARTINS DA COSTA
Data: 19/09/2024 11:07:43-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Maurílio Alves Martins da Costa
Professor Convidado

gov.br
Documento assinado digitalmente
RODRIGO GAIBA DE OLIVEIRA
Data: 19/09/2024 10:10:04-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Rodrigo Gaiba de Oliveira
Professor Convidado

Timóteo
2024

“Se estiver se sentindo desmotivado, ou sentindo que não é bom o suficiente. Incendeie seu coração. Enxugue as lágrimas e siga em frente. Quando se entristecer ou acovardar lembre-se que o fluxo do tempo nunca para, ele não vai te esperar enquanto você se afoga em tristeza”

Kyojuro Rengoku

Resumo

O presente trabalho compara qualitativa e quantitativamente métodos das bibliotecas SciPy e Pymoo aplicados ao problema otimização geométrica de uma junta adesiva a fim de reduzir as tensões normais na camada do adesivo. A solução deste tipo de problema envolve uma sequência de Análises de Elementos Finitos (FEA) à medida em que as variáveis do problema de otimização vão sendo avaliadas. Portanto, para possibilitar a integração das bibliotecas de otimização com o modelo de elementos finitos foi utilizada a interface PyMAPDL. Os métodos exatos e livres de derivadas Nelder-Mead, Powell e COBYLA foram comparados com os métodos L-BFGS-B e AG, sendo o primeiro dependente do cálculo de derivadas e o segundo, um método aproximado. Os algoritmos foram avaliados quanto à qualidade ou exatidão, tempo de execução e confiabilidade dos resultados. Os algoritmos foram avaliados em funções da literatura e no problema *benchmark*, considerando variações nos parâmetros de cada um deles. O método Nelder-Mead apresentou o melhor desempenho, atingindo o mínimo global com um chanfro de $15mm$ e uma sobreposição de $45mm$, com uma média de tempo de aproximadamente $3,0$ minutos.

Palavras-chave: otimização estrutural, métodos livres de derivadas, avaliação de desempenho, scipy, pymapdl.

Abstract

This work compares, both quantitative and qualitatively, SciPy and Pymoo library methods applied to geometric optimization problem of an adhesive joint in order to decrease normal tensions in the adhesive layer. The solution to this type of problem involves a Finite Element Analysis (FEA) sequence as the optimization problem variables are evaluated. Therefore, to enable the integration of optimization libraries with the finite element model, the PyMAPDL interface was used. The exacts and derivatives-free methods Nelder-Mead, Powell and COBYLA were compared with the L-BFGS-B and AG methods, the first being dependent on the calculation of derivatives and the second is an approximate method. The algorithms were evaluated for quality or accuracy, execution time and reliability of results. The algorithms were evaluated using functions from the literature and without benchmark problems, considering variations in the parameters of each of them. The Nelder-Mead method presented the best performance, reaching the global minimum with a $15mm$ chamfer and $45mm$ overlap, with an average time of approximately 3.0 minutes.

Keywords: structural optimization, derivative-free methods, performance evaluation, scipy, pymapdl.

Lista de Figuras

Lista de tabelas

Lista de abreviaturas e siglas

AG	Algoritmo Genético
BESO	Otimização Estrutural Evolutiva Bidirecional
COBYLA	Otimização Restrita por Aproximações Lineares
JAT	Juntas Adesivas Tubulares
L-BFGS-B	Broyden-Fletcher-Goldfarb-Shanno de Memória Limitada com Restrições Vinculadas
MDC	Modelo de Dano Coesivo
MEF	Método dos Elementos Finitos
PO	Pesquisa Operacional
POSMAT	Programa de Pós-Graduação em Engenharia de Materiais
PPL	Problema de Programação Linear
SLJ	Junta de Sobreposição Simples

Sumário

1 Introdução

O avanço nas áreas de tecnologia assume hoje o papel de alicerce no mundo, o que passa a exigir das empresas e dos meios de produção a busca rápida por alternativas e propostas para a otimização de suas tarefas, produtos e serviços (MELLO; PEREIRA, 2016).

Em termos gerais, a otimização refere-se ao processo de buscar a melhor solução possível para um problema, considerando ou não determinadas restrições e objetivos específicos. No escopo da otimização há basicamente dois tipos de segmentos: métodos exatos e aproximados (BORGES, 2013).

Enquanto os algoritmos exatos asseguram a identificação da solução ótima, seu esforço computacional cresce exponencialmente com o aumento das dimensões do problema. Por outro lado, os algoritmos aproximados foram concebidos para proporcionar soluções de qualidade, aproximando-se do valor ótimo, com um tempo de processamento mais adequado (HILLIER; LIEBERMAN, 2010; BORGES, 2013). Como resultado, esses algoritmos aproximados viabilizam a resolução de problemas de médio e grande porte (ROTHLAUF, 2011).

De acordo com Chapra e Canale (2009), equipes de engenharia devem projetar produtos e soluções que desempenhem tarefas de forma eficiente e a um custo baixo, e por isso estão recorrentemente diante problemas de otimização que equilibram eficiência e limitações. No âmbito das engenharias mecânica e de materiais, por exemplo, é comum se deparar com problemas de otimização geométrica, os quais consistem em buscar a geometria mais eficiente para um determinado objeto ou estrutura, de acordo com certas condições, com o objetivo de maximizar ou minimizar determinadas características, como a resistência mecânica, a eficiência energética, a aerodinâmica, entre outras (MUNDSTOCK, 2006).

Para aplicar um método de otimização, seja ele exato ou aproximado, a equipe de engenharia pode adotar diferentes recursos computacionais, que vão desde planilhas até algoritmos parametrizáveis implementados em bibliotecas de linguagens de programação. Para o ambiente python, as bibliotecas SciPy e Pymoo são amplamente utilizadas no contexto de programação científica e fornecem a flexibilidade necessária para a avaliação de diferentes parâmetros.

Os métodos selecionados (Nelder-Mead, Powell, COBYLA, L-BFGS-B e AG) dispõem de parâmetros que influenciam a convergência. Uma parametrização adequada pode garantir ao método qualidade ou exatidão, diminuição do tempo de execução e confiabilidade dos resultados. Essas características formam um conjunto de métricas que serão utilizadas neste trabalho para avaliar os métodos.

O problema *benchmark* selecionado para a avaliação dos dos métodos de otimização e suas parametrizações foi a otimização geométrica de uma junta adesiva de sobreposição simples, que também é um objeto de pesquisa no Programa de Pós-graduação em Engenharia de Materiais (POSMAT) do campus Timóteo (GOVEIA, 2022). Nestas estruturas, o aumento da

resistência mecânica do adesivo não garante o aumento da resistência da estrutura como um todo, sendo necessária a realização de modificações em sua geometria a fim de se promover uma distribuição mais uniforme das tensões, reduzindo a concentração de tensões normais e aumentando assim a vida útil da estrutura.

Para modelar computacionalmente a estrutura da junta adesiva sob esforço de tração, é necessário recorrer a métodos numéricos para a solução de equações diferenciais parciais, como por exemplo, o Método dos Elementos Finitos (MEF). A interface PyMAPDL, lançada em 2021 pela empresa ANSYS, possibilita a modelagem por elementos finitos diretamente no ambiente python, o que viabiliza a integração também com outras bibliotecas da linguagem (GIL, 2021), como por exemplo, a Scipy e Pymoo.

Diante deste cenário, coloca-se a seguinte pergunta de pesquisa: Quais métodos implementados nas bibliotecas SciPy e Pymoo são mais adequados para a otimização de um problema *benchmark* modelado por elementos finitos?

1.1 Objetivos

Para responder ao problema proposto, o objetivo geral deste trabalho é comparar qualitativa e quantitativamente os métodos das bibliotecas SciPy e Pymoo aplicados ao problema otimização geométrica de uma junta adesiva tendo como objetivo a redução das tensões normais.

Também, objetiva-se mais especificamente:

1. Apresentar uma análise do estado da arte sobre o uso de algoritmos exatos e aproximados na solução de problemas de otimização.
2. Apresentar as métricas de avaliação de desempenho de algoritmos de otimização exatos e aproximados.
3. Documentar e disponibilizar a infraestrutura necessária para o desenvolvimento de trabalhos que envolvam otimização estrutural e modelagem por elementos finitos com a interface PyMAPDL.
4. Avaliar o impacto da parametrização dos métodos exatos na qualidade, robustez e tempo de convergência.

2 Fundamentação Teórica

2.1 Otimização

A otimização, ou programação matemática, é uma área fundamental da pesquisa operacional (PO), que é uma disciplina que utiliza técnicas matemáticas, estatísticas e de modelagem para ajudar na tomada de decisões eficientes em organizações e processos (LEIGUS; FENERICH; MORAIS, 2009; HILLIER; LIEBERMAN, 2010).

Portanto, a otimização, no contexto da PO, refere-se ao processo de encontrar a melhor solução possível para um problema, dadas certas restrições e objetivos específicos. O objetivo é maximizar ou minimizar uma função objetivo, sujeita a um conjunto de restrições (LEIGUS; FENERICH; MORAIS, 2009).

A otimização pode ser utilizada para resolver problemas em uma ampla gama de áreas, como logística, planejamento de produção, roteamento de veículos, programação de projetos, alocação de recursos, entre outros. Ao otimizar as decisões tomadas em uma organização ou processo, é possível melhorar a eficiência, reduzir custos, aumentar a produtividade e alcançar resultados mais favoráveis (HILLIER; LIEBERMAN, 2010).

Segundo Haftka e Gurdal (1991), a formulação de um problema de otimização pode ser feita por meio do seguinte exemplo:

$$Maximizar \quad f(x_1, x_2, x_3) = 2x_1 - x_2 + 4x_3 \tag{2.1}$$

sujeito às restrições:

$$x_1 + 2x_2 + x_3 = 3 \tag{2.2}$$

$$x_2 + 2x_3 \geq 4 \tag{2.3}$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \tag{2.4}$$

Os elementos básicos que compõem este e outros problemas de otimização são os seguintes:

- **Função objetivo:** de acordo com Christensen e Klarbring (2008) e Berastegui (2020), é a função de uma ou mais variáveis de projeto que se quer otimizar, minimizando-a ou maximizando-a. No exemplo, a função objetivo é dada pela equação 2.1.

- **Variáveis de decisão:** conforme apresentado por Haftka e Gurdal (1991), são as variáveis independentes da função objetivo, as quais se deseja conhecer o valor, de forma a otimizá-la. No exemplo, a variáveis de decisão são x_1, x_2 e x_3.
- **Restrições:** segundo Haftka e Gurdal (1991), as restrições em um problema de otimização podem ser de 3 tipos:
  - **Restrições de igualdade**: este tipo de restrição estabelece condições de igualdade para determinadas grandezas, relacionando diferentes variáveis de decisão, como pode ser visto na equação 2.2.
  - **Restrições de desigualdade**: são as que definem limites superior ou inferior para determinadas grandezas, relacionando diferentes variáveis de decisão, como pode ser visto na equação 2.3.
  - **Restrições laterais ou de caixa**: são aquelas restrições de limites superior e inferior nas variáveis de decisão, como pode ser visto nas equações 2.4.

2.2 Métodos para a solução de problemas de otimização

A seleção de métodos para a solução de problemas de otimização deve levar em conta o funcionamento do método de solução, os quais podem ser classificados como determinísticos ou estocásticos, exatos ou aproximados (BERASTEGUI, 2020; KOZIEL; YANG, 2011), baseados em gradiente ou livres de derivadas, com ou sem suporte à restrições.

De acordo com Berastegui (2020), Koziel e Yang (2011), Goldbarg e Luna (2000) e (NOCEDAL; WRIGHT, 2006):

- **determinístico x estocástico:** um método determinístico retorna sempre a mesma solução dadas as mesmas configurações de entrada, enquanto um método estocástico utiliza distribuições de probabilidade geradas aleatoriamente a fim de investigar diferentes cenários de solução;
- **exato x aproximado:** um método exato retorna sempre a melhor a solução possível, enquanto um método aproximado retorna uma solução aproximada, sem a garantia de estar próxima da melhor solução;
- **baseado em gradiente x livre de derivadas:** um método método baseado em gradiente utiliza informações das derivadas da função objetivo para encontrar a solução, enquanto um método livre de derivada não depende de informações das derivadas para encontrar a solução;
- **com suporte à restrições x sem suporte à restrições:** um método com suporte à restrições é capaz de levar em conta a existência das restrições na busca pela solução, enquanto um método sem suporte à restrições não consegue lidar com essas restrições para encontrar a solução.

A Tabela 1 apresenta as características dos algoritmos selecionados baseados nas classificações discutidas.

Tabela 1 – Características dos algoritmos selecionados

Algoritmo	**Determinismo**	**Exatidão**	**Derivada**	**Restrição**
Nelder-Mead	determinístico	exato	livre de derivada	irrestrito
Powell	determinístico	exato	livre de derivada	irrestrito
COBYLA	determinístico	exato	livre de derivada	restrito
L-BFGS-B	determinístico	exato	baseado em derivada	irrestrito
AG	estocástico	aproximado	livre de derivada	irrestrito

Fonte: o autor

2.2.1 Nelder-Mead

O método do Nelder-Mead, também conhecido como método Simplex, foi proposto por John Nelder (1924 – 2010) e Roger Mead (1938 – 2015) em 1965 como uma técnica de otimização. Este método é amplamente utilizado em problemas de otimização não linear devido à sua simplicidade e eficácia (NELDER; MEAD, 1965).

O método do Nelder-Mead é especialmente útil quando as funções objetivo não são diferenciáveis ou quando as derivadas são difíceis e/ou computacionalmente caras de calcular. Ele é capaz de lidar com problemas de otimização em espaços de alta dimensão, tornando-se uma ferramenta valiosa em uma variedade de aplicações, incluindo engenharia, ciência de dados e aprendizado de máquina. O método também é eficaz para otimizar funções não lineares que possuem uma única solução ótima ou uma solução próxima o suficiente. Ele pode ser aplicado em problemas de minimização e maximização, buscando encontrar os valores dos parâmetros que otimizam uma determinada função objetivo, sujeita a restrições de caixa, se houver (LAGARIAS et al., 1998).

O método opera com um conjunto de $n + 1$ vértices que formam um simplex, um polígono convexo em espaços com dimensão n. Primeiramente, são selecionados pontos iniciais no espaço de busca (chutes), que representam as soluções candidatas. Em cada iteração, o método avalia a função objetivo em cada vértice do simplex e realiza operações de reflexão, expansão, contração e redução para explorar e atualizar a posição dos vértices (WRIGHT, 1996).

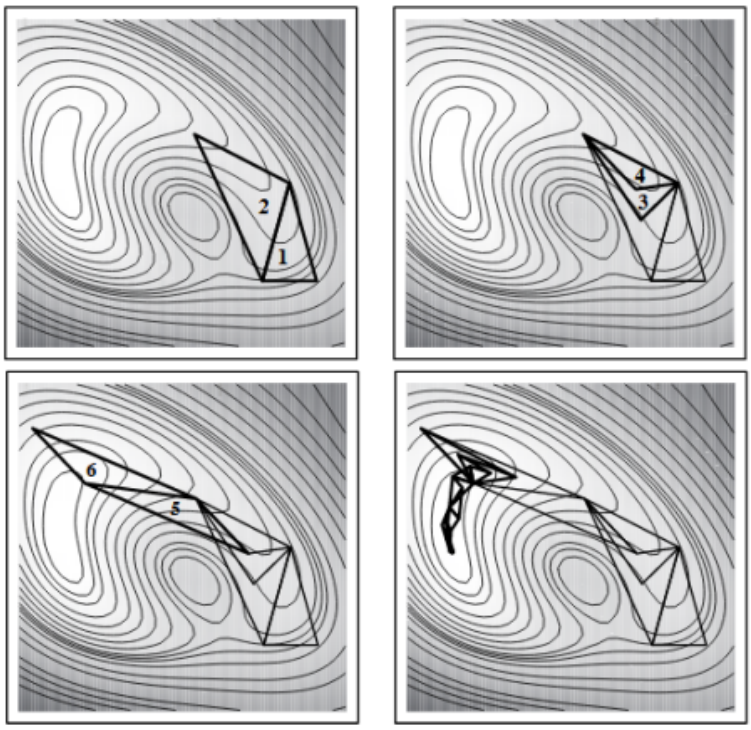
Os passos do algoritmo podem ser resumidos da seguinte forma:

1. **Inicialização:** Seleciona-se um conjunto inicial de pontos no espaço de busca para formar o simplex inicial.
2. **Avaliação:** Calcula-se o valor da função objetivo em cada vértice do simplex.
3. **Ordenação:** Ordena-se os vértices conforme os valores da função objetivo, do menor para o maior.

4. **Reflexão:** Gera-se um novo ponto refletindo o pior ponto em relação ao centro do simplex.
5. **Expansão:** Se o novo ponto gerado pela reflexão apresentar um valor da função objetivo melhor que o segundo melhor ponto, é realizada uma expansão para além desse ponto.
6. **Contração:** Se o novo ponto gerado pela reflexão não melhorar a solução, é realizada uma contração em direção ao melhor ponto.
7. **Redução:** Se nenhuma das etapas anteriores resultar em uma melhoria, todos os pontos do simplex, exceto o melhor, são reduzidos em direção ao melhor ponto.
8. **Critério de Convergência:** Repete-se o processo até que um critério de convergência seja atendido.

A Figura 1 mostra um esquema do funcionamento do algoritmo Nelder-Mead. Dado um chute inicial, um simplex é formado, delimitando a região de busca. O valor da função em cada vértice é calculado e o pior vértice é substituído por um novo, que é gerado por meio de operações de reflexão, expansão, contração e redução. Essas operações fazem o simplex caminhar, delimitando novas regiões de busca. Esse processo é repetido até que um ótimo local seja encontrado.

Figura 1 – Iterações do algoritmo Nelder-Mead



Fonte: Givens (2012)

2.2.2 Powell

O método de Powell, proposto por Michael J. D. Powell (1936 - 2015) em 1964, surgiu como uma extensão do método de rotação de eixos, introduzindo uma abordagem iterativa para encontrar o mínimo de uma função objetivo (POWELL, 1964).

Antes do método de Powell, os métodos de otimização eram principalmente baseados em gradientes ou derivadas, o que limitava sua aplicabilidade a funções diferenciáveis. O método de Powell se destacou por sua capacidade de lidar com funções não lineares e não diferenciáveis de forma eficaz. Ele é particularmente útil em problemas de otimização multivariável, nos quais pode explorar eficientemente o espaço de busca (POWELL, 1964).

Neste método define-se inicialmente uma base vetorial para o espaço de otimização trabalhado. Normalmente a base adotada é a canônica. Definida a base, explora-se sequencialmente a minimização unidimensional de cada uma destas direções. Ao encerrar este primeiro ciclo, calcula-se a direção formada pelo ponto final deste processo e o ponto inicial. Esta direção, dita conjugada, formada pela minimização da função em cada uma das direções da base adotada, substituirá a primeira da base. Explora-se novamente cada uma das direções da nova base, formando-se então outra direção conjugada neste ciclo, que substituirá então a segunda direção da base inicial (POWELL, 1964; MENDES, 2010).

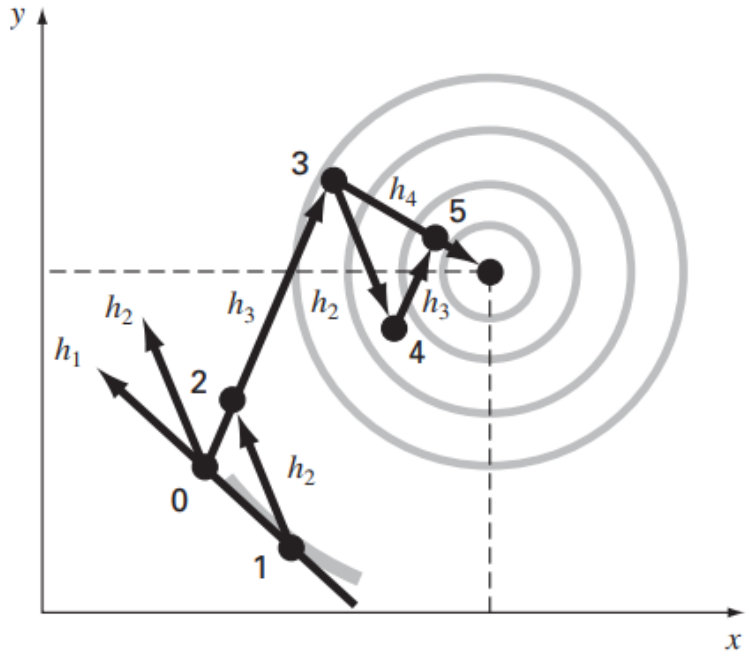
Os passos do algoritmo podem ser estruturados da seguinte forma:

1. **Inicialização:**
   1.1. Escolhe-se um ponto inicial no espaço de busca da função.
   1.2. Define-se um conjunto inicial de direções de busca que normalmente começa com as direções dos eixos (vetores unitários).
2. **Busca Unidimensional Iterativa para cada direção:**
   2.1. Realiza-se uma busca unidimensional (método da razão áurea, por exemplo), para encontrar o passo que minimiza a função.
   2.2. Atualiza-se o ponto atual adicionando o valor do passo.
3. **Atualização da Direção de Busca:**
   3.1. Calcula-se a direção do movimento total feito.
   3.2. Adiciona-se o valor do movimento total feito ao conjunto de direções de busca, substituindo uma das direções existentes. A direção substituída é aquela que contribuiu menos para a redução do valor da função.
4. **Critério de Convergência:** Repete-se o processo até que um critério de convergência seja atendido.

A Figura 2 mostra um esquema do funcionamento do algoritmo Powell. A busca se inicia no ponto 0 com direções iniciais $h1$ e $h2$. Do chute inicial, o algoritmo se move ao longo

de $h1$ até um ótimo local ser encontrado no ponto 1. Um nova busca é feita partindo do ponto 1 na direção $h2$ até encontrar um ótimo local no ponto 2, formando uma nova direção de busca $h3$ entre os pontos 0 e 2. Do ponto 2 ao longo da direção $h3$ um ótimo local é encontrado no ponto 3. Do ponto 3 ao longo da direçã $h2$ um ótimo local é encontrado no ponto 4. Do ponto 4 se chega ao ponto 5 buscando novamente ao longo da direção $h3$. Os pontos 5 e 3 foram localizados buscando na direção $h3$, a partir de dois pontos diferentes. Powell demonstrou que $h4$ (formada pelos pontos 3 e 5) e $h3$ são direções conjugadas. Assim, ao fazer uma busca a partir do ponto 5 na direção $h4$, o algoritmo encontra um ótimo local.

Figura 2 – Iterações do algoritmo de Powell



Fonte: Chapra e Canale (2009)

2.2.3 COBYLA

O método COBYLA (Constrained Optimization By Linear Approximations) é um método utilizado para resolver problemas de otimização não lineares restritos. Desenvolvido por Michael J. D. Powell (1936 - 2015) em 1994. Sua origem remonta à necessidade de resolver problemas de otimização complexos presentes em diversas áreas da ciência e engenharia (POWELL, 1994).

O COBYLA lida bem com problemas de otimização não convexas e não lineares, sendo amplamente utilizado em diversas áreas, incluindo engenharia, física, finanças e aprendizado de máquina. Sua capacidade de tratar de restrições de igualdade e desigualdade o torna uma escolha versátil para uma ampla gama de problemas de otimização (POWELL, 1994).

O método emprega uma abordagem baseada em regiões (ou raio) de busca para explorar o espaço de busca da função objetivo. Inicialmente, uma solução viável é escolhida como ponto de partida. Em seguida, é gerado um conjunto de pontos (simplex) nos quais a função objetivo é linearmente aproximada. Durante cada iteração, o COBYLA ajusta a região

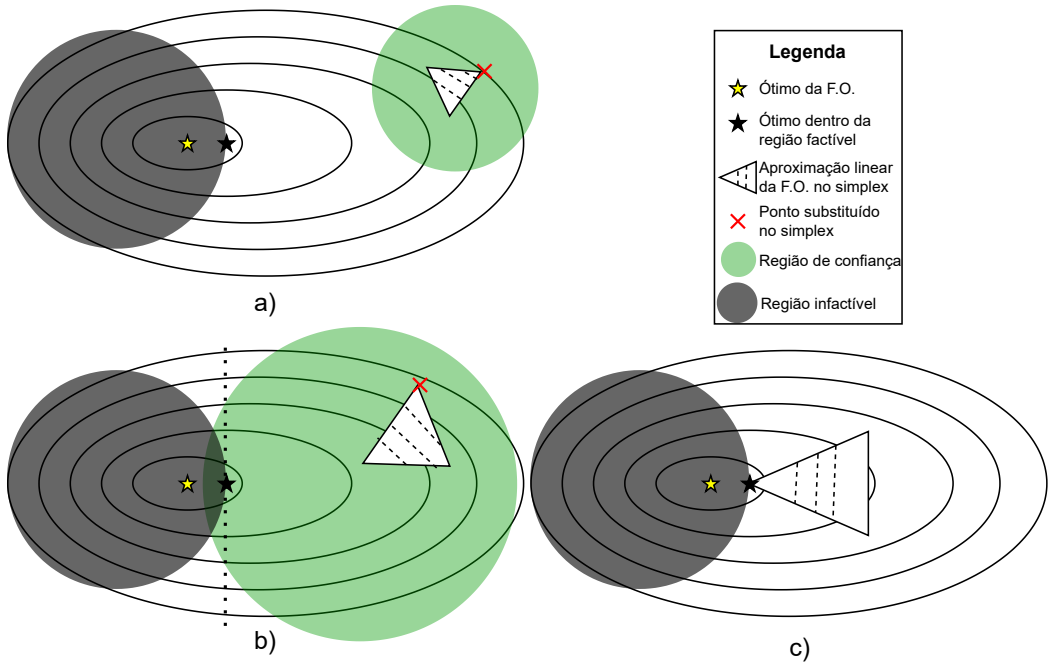
de busca ao redor do ponto atual, usando aproximações lineares da função objetivo e das restrições para determinar a direção de busca. Esses pontos são então avaliados por meio da função objetivo e novas atualizações na região de busca e no simplex são avaliadas. Este processo continua até que um critério de parada seja satisfeito, como a convergência da solução ou o número máximo de iterações seja alcançado(POWELL, 1994).

Os passos do algoritmo podem ser resumidos da seguinte forma:

1. **Inicialização:** Escolhe-se um ponto inicial. A partir desse ponto, dentro de um certo raio (região de busca ou de confiança) é definido um simplex inicial (um conjunto de pontos em torno do ponto inicial).
2. **Modelo Linear:** Realiza-se uma aproximação linear da função objetivo e das restrição dentro do simplex, obtendo-se assim um problema de programação linear (PPL).
3. **Atualização do Ponto Atual:** Resolve-se o PPL para encontrar uma direção de busca que melhora o valor da função objetivo enquanto respeita as restrições. O vértice do simplex que menos contribui para a função objetivo é então atualizado movendo-se ao longo da nova direção de busca até um novo ponto.
4. **Ajuste da região de busca:** Ajusta-se o tamanho da região de busca de acordo com o novo ponto. Se ele melhora significativamente a função objetivo e satisfaz as restrições, a região de busca pode ser expandida. Caso contrário, é reduzida.
5. **Critério de Convergência:** Repete-se o processo até que um critério de convergência seja atendido.

A Figura 3 mostra um esquema do funcionamento do algoritmo COBYLA. A partir do chute inicial é gerado um simplex (triângulo em 2D). Este simplex delimita a região de busca e dentro desta região é feita uma aproximação linear. O problema de programação linear (PPL) é então resolvido e o ponto que menos contribui para a função objetivo linearizada é substituído por um novo ponto na região de confiança de raio ρ (Figura 3.a). Assim como no algoritmo Nelder-Mead, o recálculo do novo ponto do simplex envolve operações como expansão, reflexão, contração e redução. Após a obtenção do novo simplex, a região de confiança é ajustada (por exemplo, se o novo ponto melhora a solução, a região de confiança é ampliada). O processo de linearização, solução do PPL, adequação simplex e da região de confiança é então repetido, até que se alcance um ótimo local da região factível (Figuras 3.b e 3.C).

Figura 3 – Iterações do algoritmo COBYLA



Fonte: o autor

2.2.4 L-BFGS-B

O Método de L-BFGS-B, ou Limited-memory Broyden-Fletcher-Goldfarb-Shanno with Bound Constraints, é utilizado para resolver problemas de otimização não linear com restrições de caixa. Desenvolvido como uma extensão do método BFGS, o L-BFGS-B foi proposto por Ciyou Zhu, Richard Byrd, Jorge Nocedal e Peihuang Lu nos anos 1990. O método foi concebido para superar limitações dos algoritmos de otimização clássicos, especialmente em problemas com grandes conjuntos de dados ou quando a matriz Hessiana completa é difícil ou impossível de calcular (BYRD et al., 1995).

As principais características do método L-BFGS-B incluem sua eficiência computacional e sua capacidade de lidar com problemas de otimização não linear restrita, com as variáveis sujeitas apenas a limites superiores e inferiores. Ele é particularmente útil em aplicações onde a memória é limitada, pois utiliza uma abordagem de memória limitada para aproximar a inversa da Hessiana. Além disso, o L-BFGS-B é robusto em problemas com muitas variáveis e pode lidar com funções objetivo não diferenciáveis de forma aproximada (BYRD et al., 1995).

Para o método funcionar, é necessário fornecer uma estimativa inicial para as variáveis de otimização. Em seguida, o algoritmo itera por uma sequência de passos, nos quais calcula uma direção de descida utilizando a informação armazenada nas últimas iterações. Esta direção é obtida por meio de uma combinação da direção de Newton e da direção de descida do gradiente. Posteriormente, uma busca linear é realizada ao longo desta direção para determinar o tamanho do passo que minimiza a função objetivo dentro dos limites impostos. Esse processo continua até que uma condição de parada seja satisfeita. Em cada iteração, o método

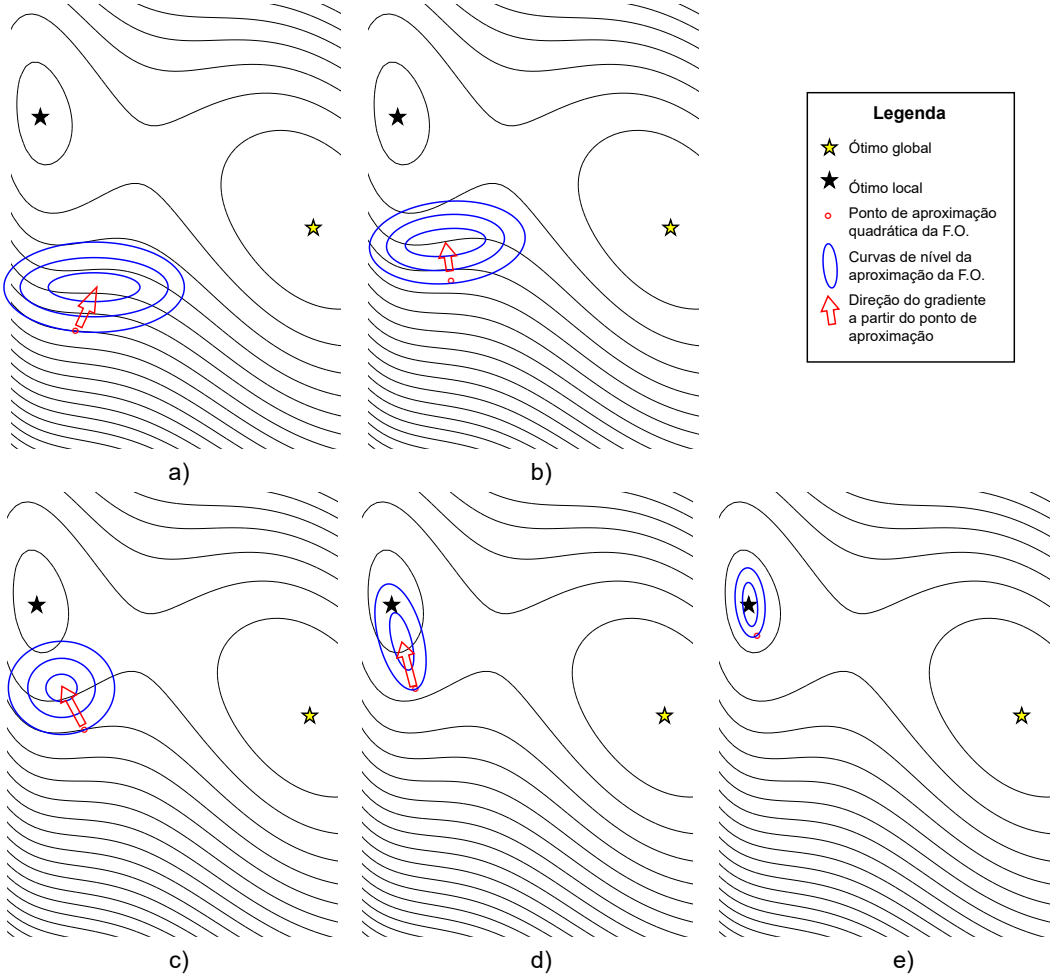
atualiza uma estimativa da inversa da Hessiana usando um conjunto mínimo de informações armazenadas nas iterações anteriores, garantindo eficiência computacional em problemas de grande escala (MORALES; NOCEDAL, 1997).

Os passos do algoritmo podem ser estruturados da seguinte forma:

1. **Inicialização:**
   1.1. Escolhe-se um ponto inicial e define-se os limites para cada variável.
   1.2. Inicializa-se os parâmetros, como o número máximo de iterações, tolerância e quantidade de informações a serem armazenadas para a estimativa da matriz Hessiana.
2. **Atualização da Direção de Busca:**
   2.1. Calcula-se o gradiente da função objetivo no ponto atual.
   2.2. Aproxima-se a inversa da matriz hessiana utilizando as informações dos gradientes anteriores.
   2.3. Estima-se a nova direção de busca a partir da hessiana aproximada.
3. **Determinação do Passo:**
   3.1. Aplica-se uma busca linear para determinar o tamanho do passo ao longo da nova direção de busca, respeitando as restrições de caixa.
4. **Atualização do Ponto:**
   4.1. Atualiza-se o ponto atual x movendo-o de acordo com o tamanho de passo calculado, ao longo da direção de busca atual.
5. **Critério de Convergência:** Repete-se o processo até que um critério de convergência seja atendido.

A Figura 4 mostra o processo iterativo do algoritmo BFGS. Em um ponto inicial (chute) é feita uma aproximação da matriz hessiana, representada pelo paraboloide que passa pelo ponto inicial e tem a mesma curvatura da função objetivo no ponto. A hessiana aproximada é então minimizada por meio de busca linear na direção do gradiente do ponto inicial (Figura 4.a). O ponto ótimo encontrado passa a ser o novo ponto de aproximação, e o processo é repetido até que o ótimo da função objetivo original seja atingido (Figura 4.b em diante).

Figura 4 – Iterações do algoritmo BFGS



Fonte: o autor

2.2.5 Algoritmos Genéticos (GA)

Os Algoritmos Genéticos são uma técnica de otimização e busca baseada nos princípios da genética e seleção natural, proposta por John Holland na década de 1960 na Universidade de Michigan. Holland e seus colaboradores desenvolveram o conceito de AGs como uma forma de incorporar os mecanismos de evolução natural em sistemas computacionais, visando a resolução de problemas complexos. O trabalho inicial de Holland culminou na publicação do livro “Adaptation in Natural and Artificial Systems” em 1975, onde formalizou os conceitos de algoritmos genéticos. Desde então, os AGs têm sido aplicados em diversos campos, desde engenharia até economia, provando ser uma ferramenta eficaz para encontrar soluções ótimas em problemas onde métodos tradicionais falham (GOLDBERG, 1989).

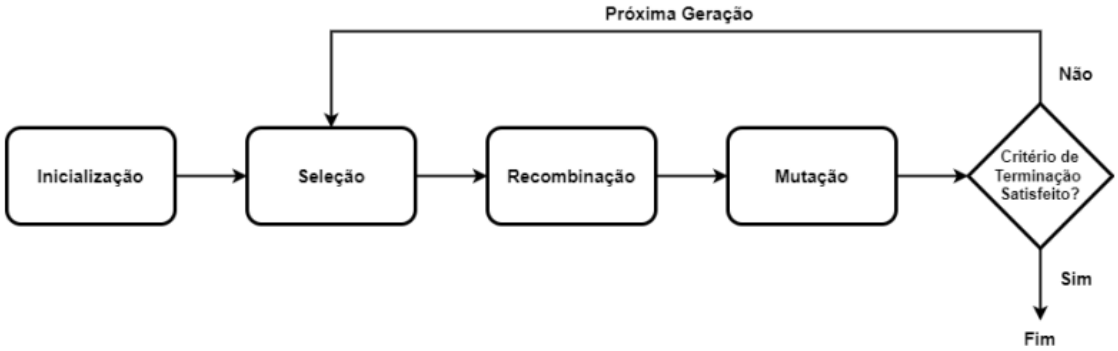
Os algoritmos genéticos são caracterizados por sua robustez e versatilidade, sendo capazes de resolver uma ampla gama de problemas de otimização. Uma de suas principais características é a habilidade de realizar buscas em espaços de soluções complexos e extensos, onde métodos analíticos e numéricos convencionais podem não ser eficazes. Os AGs

são particularmente úteis em problemas de otimização global, problemas de programação não linear, otimização de múltiplos objetivos, e problemas onde as variáveis de decisão são discretas ou mistas. Sua aplicabilidade estende-se por áreas como design de engenharia, otimização de redes, finanças, e bioinformática, oferecendo soluções para problemas que requerem uma exploração extensiva do espaço de busca (COELLO, 2002). A Figura 5 ilustra o funcionamento de um AG genérico.

Os passos do algoritmo podem ser estruturados da seguinte forma:

1. **Inicialização:** Gera-se uma população inicial de indivíduos (soluções) de forma aleatória ou por meio de algum critério pré-definido.
2. **Avaliação:** Avalia-se cada indivíduo da população com base em uma função de aptidão (fitness), que quantifica o quão bem ele resolve o problema.
3. **Seleção:** Seleciona-se indivíduos para reprodução baseados em sua aptidão, utilizando métodos como seleção por torneio ou roleta.
4. **Cruzamento (Recombinação):** Cruza-se pares de indivíduos para formar descendentes, combinando partes de suas soluções.
5. **Mutação:** Aplica-se pequenas mudanças aleatórias aos descendentes para manter a diversidade genética da população.
6. **Substituição:** Substitui-se a população atual pela nova geração de indivíduos (descendentes), integral ou parcialmente, dependendo do esquema de substituição utilizado.
7. **Critério de Convergência:** Repete-se o processo até que um critério de convergência seja atendido.

Figura 5 – Modelo de um AG



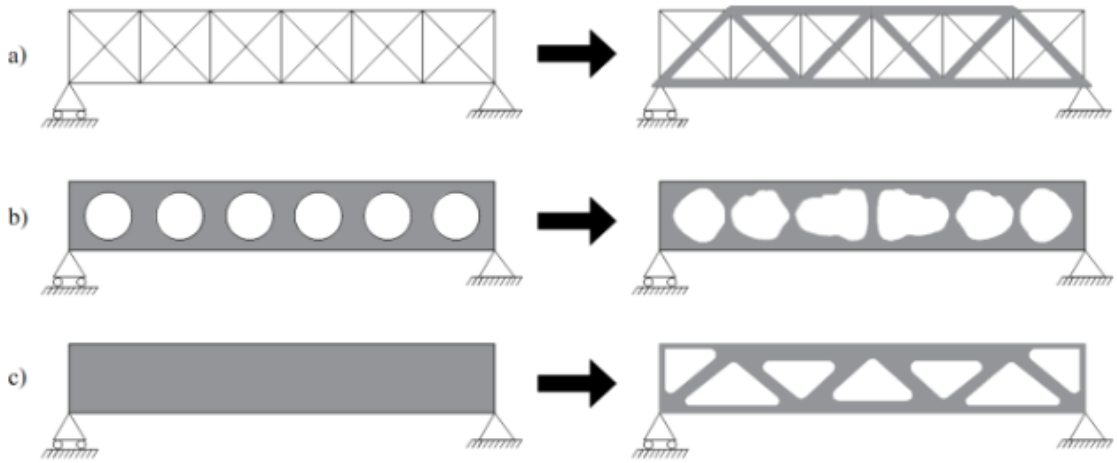
Fonte: Araujo (2022)

2.3 Otimização geométrica

De acordo com Mundstock (2006) no campo da engenharia, a otimização estrutural é uma área de grande interesse, pois suas aplicações buscam soluções para redução de cus-

tos, material e tempo nos projetos, especialmente para componentes e sistemas que requerem uma relação crucial entre peso e desempenho, segurança e confiabilidade. Existem três grandes sub-áreas que englobam a otimização estrutural (Figura 6):

Figura 6 – Tipos de otimização: (a) Paramétrica; (b) Forma; (c) Topológica

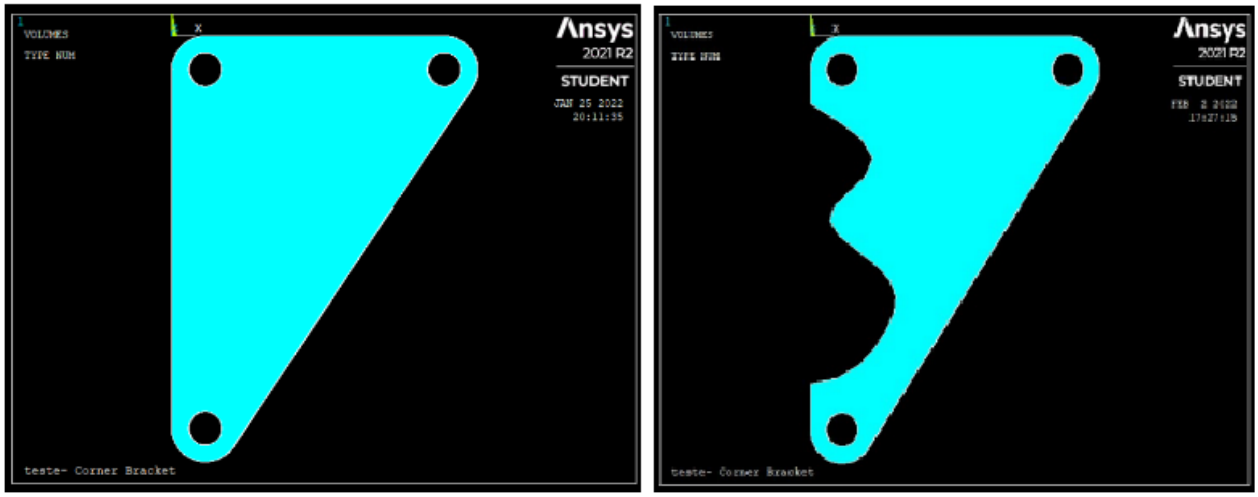


Fonte: (FERNANDES, 2021)

- Otimização paramétrica: trabalha alterando os parâmetros de uma estrutura, como, por exemplo, o diâmetro de uma viga.
- Otimização de forma: otimiza o contorno da estrutura.
- Otimização topológica: altera-se a topologia da estrutura (maneira como os pontos de um conjunto estão distribuídos e conectados, ou não, entre si). Introduzir vazios em uma placa para diminuir o gasto de material, é um exemplo de otimização topológica.

A otimização geométrica (ou otimização de forma), segundo Pedersen (1973), visa determinar a forma ótima dos contornos de estruturas, sendo aplicada, por exemplo, na adequação da geometria de sólidos para a redistribuição das tensões e redução de massa e na modificação do posicionamento dos nós de treliças. A Figura 7 mostra uma modificação na geometria de uma mão francesa a fim de reduzir a quantidade de material, mantendo as mesmas propriedades mecânicas da estrutura.

Figura 7 – Otimização de forma



a) Modelo Refência b) Modelo Otimizado

Fonte: Adaptado de Araujo (2022)

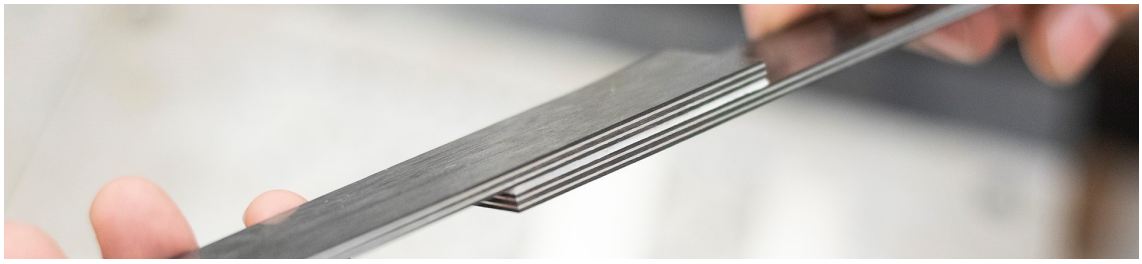
Na otimização geométrica, a formulação do problema não permite a introdução ou remoção dos elementos e nem dos nós que constituem a estrutura (VELEZ, 2015).

2.4 Junta Adesiva de sobreposição simples

As juntas adesivas são amplamente utilizadas em várias indústrias, como automotiva, aeroespacial, eletrônica e construção, devido às suas vantagens em relação a outros métodos de união, como soldagem ou fixação mecânica (OCANA et al., 2015).

Uma junta adesiva pode ser definida como a união de duas ou mais superfícies por meio de uma substância adesiva projetada para criar uma ligação forte e durável entre as partes (REIS, 2022; GOVEIA, 2022). Embora existam diferentes configurações de juntas adesivas, a sobreposição simples dos aderentes é a mais comumente utilizada na indústria, principalmente em razão da facilidade de fabricação, manutenção e inspeção, em relação a outros tipos de juntas (GOVEIA, 2023). A Figura 8 mostra uma junta adesiva de sobreposição simples.

Figura 8 – Junta de sobreposição simples para ensaio de tração

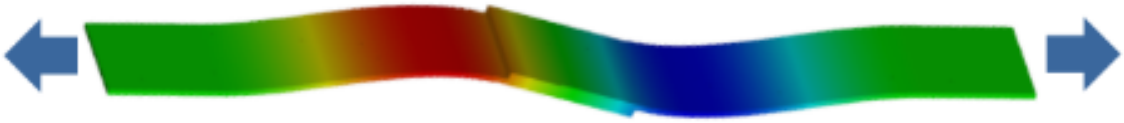


Fonte: Inegi (2020)

Um dos problemas experimentados no uso de juntas de sobreposição simples é a baixa resistência à tensões de delaminação (Tensões normais, isto é, na direção Y). Conforme

mostra a Figura 9), como não há simetria no centro da junta, ao ser submetida a um esforço de tração (setas azuis), ocorre o surgimento de um momento fletor (rotação) na região de sobreposição, fazendo com que surjam tensões de delaminação no adesivo e assim o descolamento do mesmo (GOVEIA, 2023).

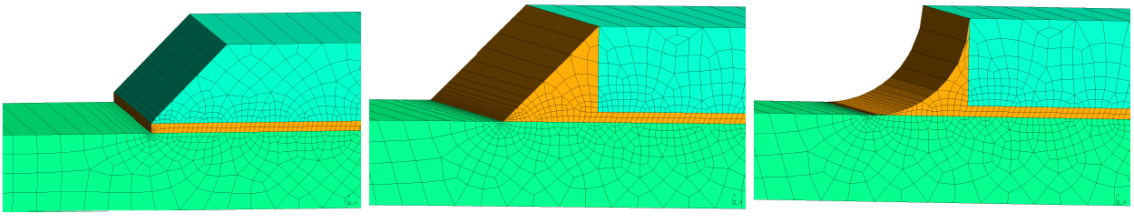
Figura 9 – Deslocamento (amplificado) em uma junta de sobreposição simples



Fonte: Goveia (2023)

A partir da identificação deste comportamento, algumas modificações podem ser realizadas no projeto da junta a fim de reduzir a assimetria na região de sobreposição e com isso, reduzir o surgimento das tensões de delaminação, contribuindo para o aumento da vida útil da estrutura. A Figura 10 mostra três possibilidades de modificação no projeto de uma junta, sendo a primeira por meio da inclusão de chanfros nos aderentes e outras duas por meio da conformação do excesso (filete) de adesivo.

Figura 10 – Possíveis modificações na geometria da junta adesiva

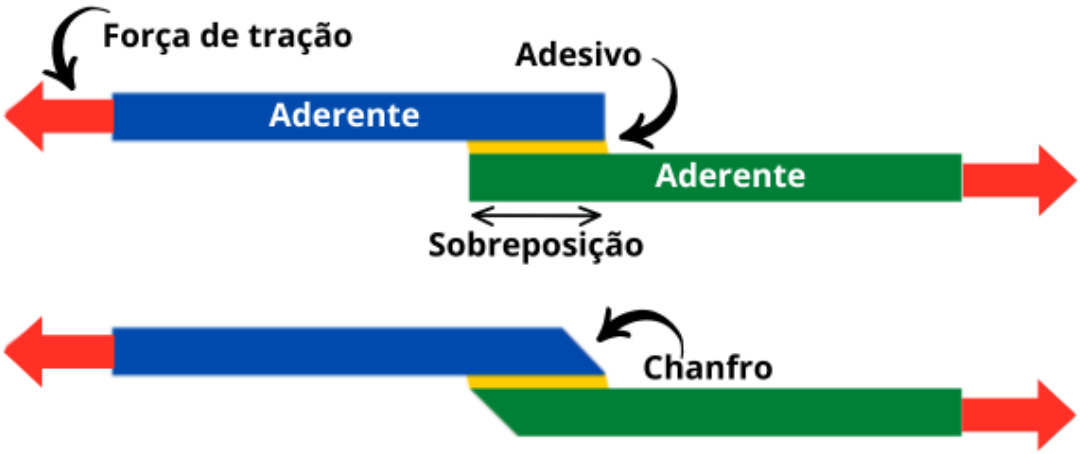


(a) Chanfro no aderente (b) Filete do adesivo (c) Raio do filete

Fonte: Goveia (2022)

A Figura 11 apresenta o esquema da junta adesiva a ser otimizada. Nela é possível observar os dois parâmetros de projeto utilizados como variáveis de decisão do problema de otimização: comprimento da sobreposição ângulo do chanfro.

Figura 11 – Esquema 2D da junta de sobreposição simples



Fonte: Adaptado de Goveia (2022)

2.4.1 Modelagem computacional

Para representar computacionalmente a junta adesiva e as tensões impostas por uma força trativa é necessário representar as relações entre tensão e deformação para cada material da estrutura.

Uma simplificação do problema consiste em considerar que os materiais estarão sujeitos apenas a pequenas deformações e por isso uma relação constitutiva linear pode ser utilizada. A Lei de Hooke generalizada define o comportamento elástico-linear do material e é apresentada na Equação 2.5. Na equação, σ é o tensor de tensões de (ou tensor de Cauchy), C é o tensor constitutivo que define o comportamento do material e ε é o tensor de deformações (SCHON, 2013). Os índices i, j, k e l indicam direções no espaço, e portanto, podem assumir os valores x, y e z.

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl} \tag{2.5}$$

O tensor C de quarta ordem é definido a partir das constantes E e ν que são respectivamente o módulo de elasticidade e o coeficiente de Poisson de cada material. Os tensores σ e ε são calculados pelo método dos elementos finitos, partindo-se das propriedades dos materiais (C) e das condições de contorno, as quais são a carga aplicada e as restrições de mobilidade da estrutura em determinados pontos (SCHON, 2013).

2.4.2 Arquitetura de comunicação do PyMAPDL

Para que a otimização estrutural seja possível é necessário modelar e resolver o modelo de elementos finitos inúmeras vezes, para os diferentes valores de variáveis de decisão obtidos no processo de otimização, até que se chegue a uma combinação ótima. Este processo exige tanto eficiência computacional quanto flexibilidade para a escolha de tecnologias

e customização do método de otimização. O PyMAPDL é um recurso que provê ambos atributos.

O PyMAPDL é uma integração do software de modelagem Ansys Mechanical com o ambiente de desenvolvimento em python. Com esse novo recurso, através do comando abaixo, é possível ter o Mechanical integrado com o python (PYANSYS, 2023).

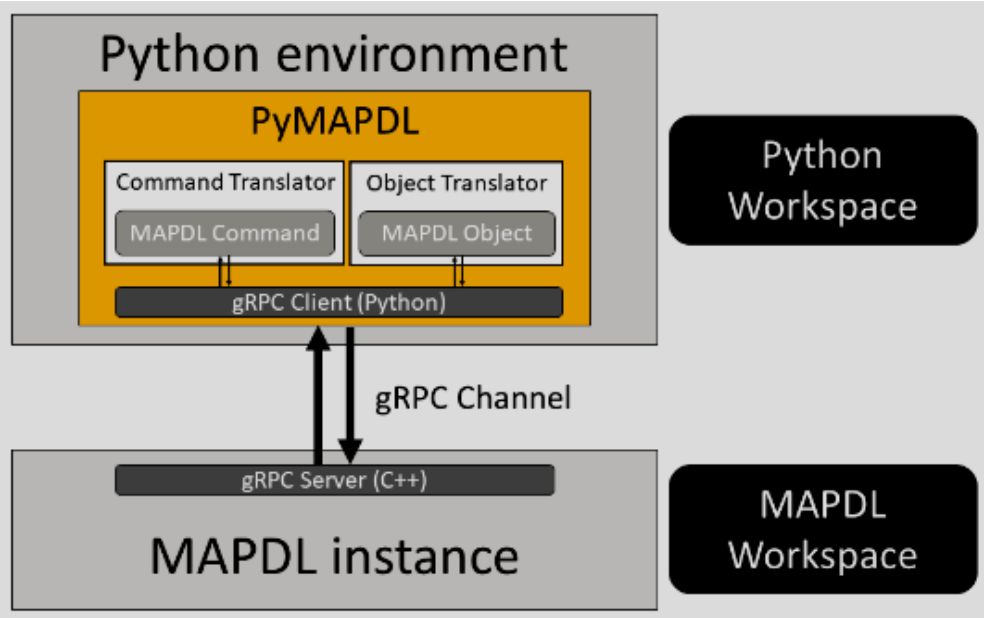
```
1    pip install ansys-mapdl-core
```

O pacote `ansys-mapdl-core` apresenta uma interface ao python para acionar o software que gerencia o envio de comandos APDL de baixo nível, enquanto troca dados por meio de interfaces gRPC (Chamadas de Procedimento Remoto do Google) de alto desempenho. O gRPC é usado para estabelecer conexões seguras para que um aplicativo cliente possa chamar métodos diretamente em uma instância MAPDL potencialmente remota como se fosse um objeto local (PYANSYS, 2023).

Isto, juntamente com o uso de formatos de transmissão binários, favorece um melhor desempenho. Usando gRPC, o PyMAPDL pode converter instruções python em comandos APDL que podem então ser transmitidos para uma instância MAPDL em execução em qualquer lugar (PYANSYS, 2023).

A Figura 12 mostra o mecanismo de comunicação entre os processos do python e MAPDL. O PyMAPDL consiste no encapsulamento dos comandos APDL por meio de classes e funções em python. Quando um método ou função é invocado, é gerado (ou traduzido para) um comando APDL, o qual é enviado para a instância do MAPDL via gRPC. Este comando é lido e executado no processo do Mechanical e o retorno é enviado de volta para a instância do python. Apesar do gRPC ser uma tecnologia eficiente (NETO, 2022), ocorre uma intensiva comunicação entre os processos, uma vez que os comandos são traduzidos individualmente, e não o programa todo.

Figura 12 – Arquitetura simplificada do PyMAPDL



Fonte: (PYANSYS, 2023)

2.5 Comparação de Algoritmos de Otimização

De acordo com Beiranvand, Hare e Lucet (2017), à medida que o número de métodos de otimização e implementações desses métodos aumentou, os pesquisadores têm realizado estudos comparativos para avaliar seu desempenho. Quando bem feitos, tais estudos podem ser de grande valor para ajudar os usuários finais a escolher o método de otimização mais adequado para seus problemas. Tais estudos são geralmente chamados de avaliação de *benchmark*.

No sentido mais geral, *benchmarking* é a comparação de um ou mais produtos a um produto padrão industrial mediante uma série de métricas de desempenho. No caso de algoritmos de otimização, em uma avaliação benchmark, os produtos são as implementações específicas de determinados métodos, e as métricas de desempenho são geradas pela execução das implementações em uma série de problemas de teste. Se um algoritmo for executado mais rápido, usar menos memória e retornar um valor final de função melhor, em todos os problemas possíveis, então pode ser considerado a melhor alternativa (BEIRANVAND; HARE; LUCET, 2017).

Em geral, as medidas de desempenho de um algoritmo se enquadram em três categorias: eficiência, confiabilidade e qualidade da resposta. A eficiência de um algoritmo de otimização refere-se ao esforço computacional necessário para obter uma solução. A confiabilidade é definida como a capacidade do algoritmo de “ter um bom desempenho” em uma ampla gama de problemas de otimização. (BEIRANVAND; HARE; LUCET, 2017). A qualidade da resposta pode ser definida como distância entre a solução obtida com a melhor solução possível, quanto menor é essa distância, melhor é a solução.

Na programação matemática, existem duas medidas principais de eficiência, o número de avaliações fundamentais e o tempo de execução (BEIRANVAND; HARE; LUCET, 2017).

O termo avaliação fundamental é usado para se referir a qualquer sub-rotina chamada pelo algoritmo para obter informações sobre o problema de otimização. O número de avaliações fundamentais é uma métrica importante na comparação de algoritmos de otimização, pois reflete a eficiência computacional e o desempenho de um algoritmo na resolução de um problema. Um algoritmo que requer menos avaliações fundamentais para encontrar uma solução aceitável é geralmente considerado mais eficiente do que um algoritmo que requer um número maior de avaliações. Isso ocorre porque a avaliação de uma função objetivo muitas vezes pode ser computacionalmente cara, especialmente em problemas complexos. Além disso, a quantidade de avaliações fundamentais também está relacionada ao tempo de execução de um algoritmo, pois mais avaliações geralmente significam mais tempo de processamento (BEIRANVAND; HARE; LUCET, 2017).

O tempo de execução é outra métrica tradicionalmente utilizada na comparação de algoritmos de otimização. Ela se refere ao tempo que um algoritmo leva para encontrar uma solução para o problema de otimização em questão. O tempo de execução é uma medida importante, pois influencia diretamente a eficiência e a utilidade prática de um algoritmo de otimização (BEIRANVAND; HARE; LUCET, 2017).

Neste trabalho, a principal medida de eficiência utilizada será o tempo de execução. Mas para analisar melhor a qualidade e a confiabilidade dos resultados encontrados na execução dos métodos, o erro absoluto (Equação 2.6) também será adotado para a avaliação dos resultados.

De acordo com Chapra e Canale (2005), o erro absoluto fornece uma indicação direta da precisão da estimativa em relação ao valor verdadeiro, independentemente da direção do erro. Quanto menor o erro absoluto, mais próxima a estimativa está do valor verdadeiro.

$$Erro_{absoluto} = |min_{exato} - min_{encontrado}| \tag{2.6}$$

3 Trabalhos Relacionados

Conforme apresentado na Seção 2.2, a escolha do método para a solução de um problema de otimização é determinante para se alcançar um resultado que atenda às expectativas do projeto. Isso ocorre, pois cada método possui pressupostos em relação ao problema a ser resolvido e também uma relação de compromisso entre as principais medidas de desempenho: eficiência, confiabilidade e qualidade da resposta.

Portanto, para selecionar um solucionador para um determinado problema de otimização, deve-se comparar os métodos candidatos levando em consideração as diferentes métricas de desempenho, como mostram os trabalhos a seguir.

Schälte, Stapor e Hasenauer (2018) compararam 18 métodos de otimização livres de derivadas em problemas clássicos e na estimativa de parâmetros de modelos biológicos, concluindo que a escolha do método depende das características do problema. Métodos baseados em gradiente, como FMINCON, mostraram maior robustez em modelos biológicos, enquanto métodos globais como o Algoritmo Genético exigiram mais avaliações. Esquemas híbridos, que combinam buscas globais sem derivadas com buscas locais baseadas em derivadas, foram considerados potencialmente mais eficazes.

Firmino (2019) implementou métodos exatos e aproximados para resolver o Problema de Recuperação de Contêineres (PRC), visando otimizar a sequência de operações de um guindaste. O autor concluiu que os algoritmos exatos não foram capazes de resolver o problema para instâncias reais com mais de 19 contêineres. Entre os métodos avaliados, o RGRASP_MNI obteve os melhores resultados, tanto em termos de número de realocações quanto de tempo de operação.

Souza (2019) propôs uma comparação entre métodos exatos e aproximados na obtenção dos valores otimizados da corrente de pick-up e do múltiplo de tempo em relés relés de sobrecorrente direcional. O autor comparou os métodos exatos Nelder-Mead e Quase-Newton (BFGS) com os métodos aproximados AG e PSO. Após analisar os resultados, concluiu-se que todos os métodos analisados foram eficientes, garantindo a coordenação dos relés direcionais com ajustes otimizados.

Guio (2021) compara um algoritmo de programação linear inteira (PLI) com heurística na solução de um problema sequenciamento de movimentações para a retirada de bobinas de aço armazenadas em um pátio com empilhamento em três camadas. O modelo PLI foi adotado apenas para soluções ótimas em pequenas instâncias e a heurística para casos maiores. Guio (2021).

Soritz, Moser e Gruber-Wölfler (2022) compararam algoritmos de otimização livres de derivadas em um benchmarking inicial e na otimização de uma reação química de acoplamento cruzado de Suzuki-Miyaura em fluxo contínuo. Os métodos híbridos ampgo-sbplx e dual-annealing-bobyqa tiveram melhor desempenho médio, superando algoritmos tradicionais

como Nelder-Mead e SNOBFIT. Algoritmos locais, como Subplex e BOBYQA, mostraram rápida convergência para regiões locais, enquanto os métodos globais nos esquemas híbridos ajudaram a evitar tais ótimos locais e direções erradas causadas por ruído.

Nunes (2022) desenvolveu uma modelagem para otimizar a alocação de Policiais Penais da Superintendência dos Serviços Penitenciários (SUSEPE) utilizando métodos exatos (CPLEX e CBC) e uma heurística baseada no algoritmo GRASP. Os resultados indicaram que a meta-heurística foi promissora na otimização da primeira função objetivo, apresentando GAPs reduzidos em comparação aos solvers exatos, tanto comercial quanto aberto, na maioria dos casos. Contudo, na rotina de re-otimização, os métodos exatos superaram a meta-heurística, mostrando-se mais eficazes em melhorar a satisfação após uma primeira otimização.

Stumpf, Götz e Leon (2023) propõem um framework para otimização de elastômeros reforçados com fibras, destacando as limitações de métodos baseados em gradientes e globais na orientação das fibras, devido a riscos de estagnação em mínimos locais ou altos custos computacionais. Eles avaliaram a eficiência do algoritmo local livre de derivadas COBYLA, que mostrou robustez na minimização da densidade de energia de deformação, alcançando a mesma solução ótima independentemente das condições iniciais. Na minimização de tensões, o COBYLA não obteve os mesmos resultados, mas os autores sugerem que ele ainda é uma alternativa viável, especialmente se combinado com estratégias híbridas que integrem buscas locais e globais.

No contexto da engenharia mecânica, a otimização desempenha um papel importante pois envolve a busca por soluções que melhorem o desempenho, eficiência e custo de sistemas e componentes mecânicos. O trabalho de Araujo (2022) foi desenvolvido no CEFET-MG, campus Timóteo, com o objetivo de otimizar a geometria de uma mão francesa utilizando o algoritmo evolucionário NSGA-II. Uma proprosta do trabalho, além da otimização, é apresentar uma arquitetura de computação para integrar a linguagem APDL (Ansys Parametric Design Language), utilizada no software Mechanical, com o ambiente de desenvolvimento python.

Outra pesquisa desenvolvida no campus, no Programa de Pós-Graduação em Engenharia de Materiais (POSMAT) é a de seleção de Materiais visando o aprimoramento de propriedades de juntas adesivas. Goveia (2023) compara, por meio de simulação computacional, a distribuição de tensões e a rigidez axial em diferentes configurações de juntas, combinando material dos aderentes, material adesivo e a geometria da estrutura. A junta de compósito unida com adesivo rígido e acabamento de filete com ângulo de $7,5^\circ$ apresentou o melhor equilíbrio entre tensão máxima e rigidez axial, reduzindo a tensão máxima de delaminação de $142MPa$ para $1,457kPa$ e apresentando uma rigidez axial $78,5\%$ maior que o segundo colocado. Um trabalho futuro especificado nesta pesquisa é a otimização geométrica das juntas adesivas, cujo efeito, combinado com a seleção de materiais, tem o potencial de melhorar consideravelmente as propriedades da estrutura.

Almeida (2020) avaliou a influência da conformação de filetes de adesivo, com diferentes ângulos, nas extremidades de ligação das juntas adesivas tubulares (JAT) por meio da aplicação de três adesivos diferentes: Araldite® AV138, Araldite® 2015 e SikaForce® 7752. Por meio de um Modelo de Dano Coesivo (MDC) observou-se que a combinação do adesivo

AV138 com filete de $7,5°$ e sobreposição de $40mm$ levou à maximização da resistência da junta.

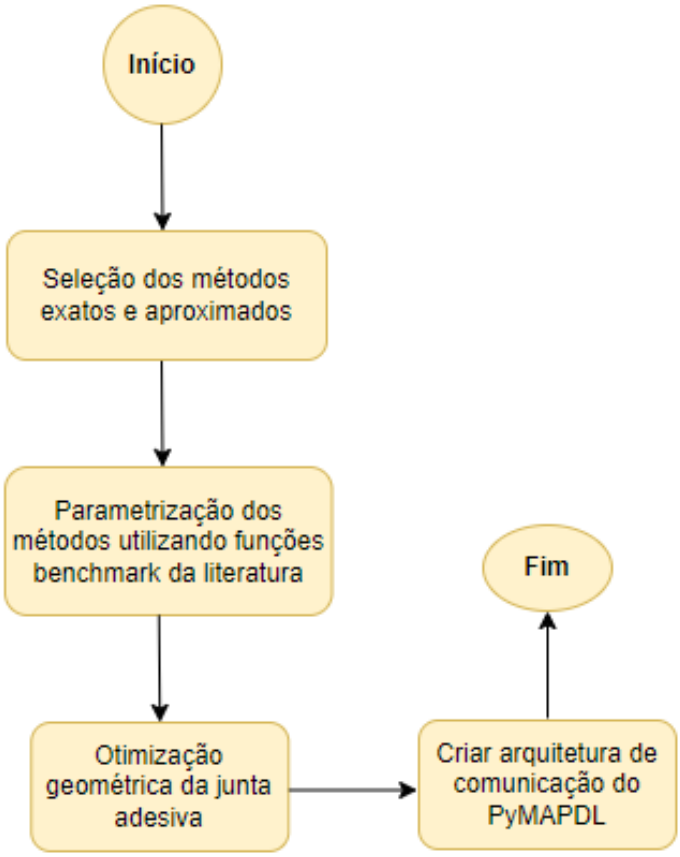
Oliveira (2021) estudou o comportamento de juntas coladas com cianoacrilato e epóxi, visando otimizar os parâmetros de fabricação para maximizar a resistência mecânica da estrutura. Utilizando adesivos Araldite (Epóxi) e Axston (Cianoacrilato) em liga de alumínio, foram testados diferentes comprimentos de sobreposição e espessuras de adesivo. Concluiu-se que um maior comprimento de sobreposição aumenta a força de ruptura da junta, mas diminui a tensão suportada devido à distribuição das tensões. Entre as combinações testadas, o adesivo Epoxy com sobreposição de $12,5mm$ apresentou o maior valor de força de ruptura, sendo considerado a melhor combinação.

Fernandes (2021) estudou do comportamento estrutural das juntas sobrepostas coladas (SLJ) e realizou simulações computacionais, por meio de MEF (Método dos Elementos Finitos) e e do algoritmo BESO (Bidirectional Evolutionary Structural Optimization) para otimizar o consumo de material adesivo e a distribuição das tensões na estrutura. Os resultados do estudo demonstraram uma concentração de material adesivo nas regiões das extremidades de juntas, uma distribuição de tensões mais uniforme e a possibilidade do aproveitamento do material de forma mais eficiente. Além disso, foi constatado que aplicação de um processo de otimização é uma ferramenta de análise estrutural que viabiliza a obtenção de uma configuração capaz de oferecer melhor aproveitamento dos materiais sob múltiplos aspectos.

4 Procedimentos Metodológicos

Neste capítulo, são apresentados o processo de seleção de métodos de otimização e das funções de estudo, os testes realizados para a análise dos algoritmos selecionados e a parametrização do modelo de otimização geométrica do problema benchmark. A Figura 13 mostra os procedimentos metodológicos adotados no desenvolvimento deste trabalho.

Figura 13 – Fluxograma dos procedimentos metodológicos



Fonte: o autor

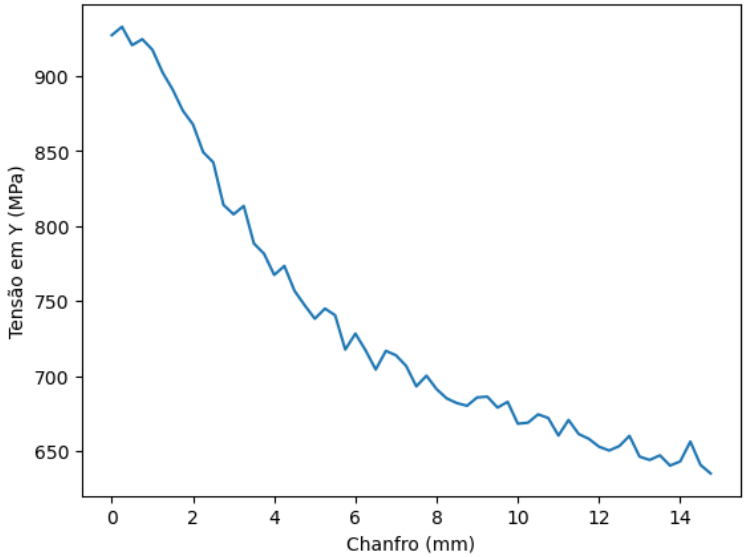
Todas as implementações e parametrizações foram feitas utilizando a ferramenta da Microsoft, Visual Studio Code, no sistema operacional Windows 10 Home Single Language, 64 bits. A máquina usada tem processador Intel Core i5-8265U 1.60 GHz, 8 GB de memória RAM, 1 TB de espaço de armazenamento em HD e 240 GB de espaço de armazenamento em SSD. A linguagem de implementação utilizada foi o python 3.

4.1 Seleção dos métodos de otimização

Serão aplicados métodos de otimização para minimizar uma função objetivo obtida a partir da modelagem numérica de uma junta adesiva pelo método dos elementos finitos. Dado

o caráter numérico, a função objetivo, apesar de contínua, não apresenta derivadas contínuas e, portanto, não é suave. Assim, os métodos de otimização baseados em gradiente não são os mais adequados. A Figura 14 mostra um exemplo de uma função objetivo “Tensão máxima de delaminação” em decorrência da variação do ângulo do chanfro.

Figura 14 – Função objetivo: Tensão máxima de delaminação



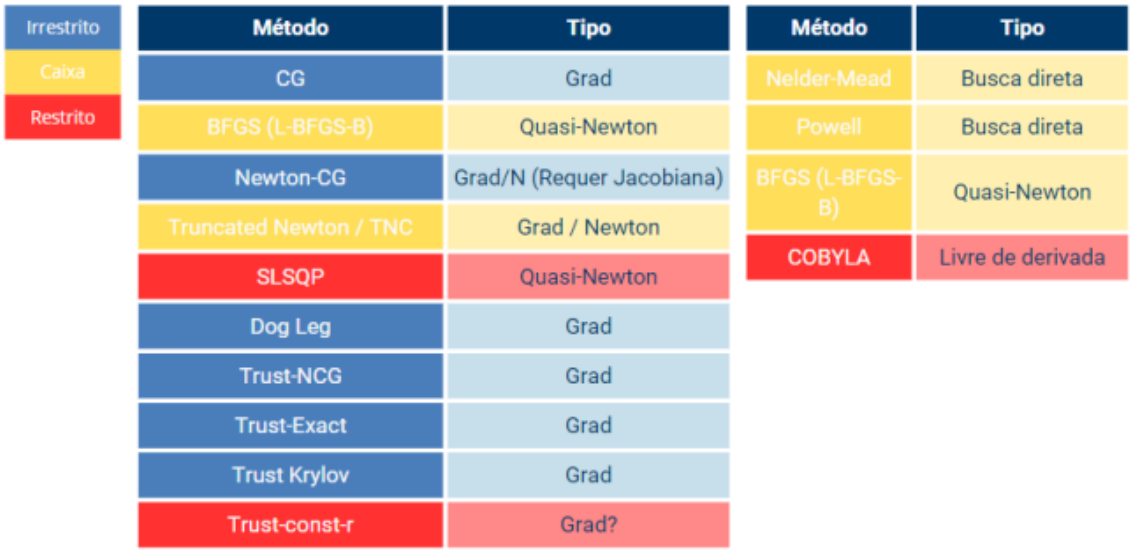
Fonte: o autor

Na Figura 15 podemos observar todos os métodos de otimização disponíveis na biblioteca SciPy. Como a função objetivo é ruidosa, buscou-se na biblioteca métodos que resolvem este tipo de problema, desse modo, os métodos baseados em gradiente foram inicialmente preteridos. A documentação do SciPy também recomenda os métodos Powell e Nelder-Mead (seções 2.2.2 e 2.2.1) para situações onde não se há conhecimento do gradiente da função e quando a função apresenta ruídos e/ou descontinuidades (VAROQUAUX, 2017). Além desses fatores, ambos os métodos possuem restrições de caixa, atendendo assim a formulação do problema.

Nesse mesmo contexto, o COBYLA (seção 2.2.3) foi selecionado por ser um método de aproximação linear que também não necessita de informações de gradiente. O método se diferencia do Powell e Nelder-Mead ao trabalhar com todos os tipos de restrições.

O método L-BFGS-B (seção 2.2.4) foi selecionado porque é baseado em gradiente, se diferenciando dos demais métodos e servindo de contraexemplo, já que teoricamente terá um desempenho inferior quando aplicado a uma função não suave (Figura 14).

Figura 15 – Métodos de otimização da Scipy

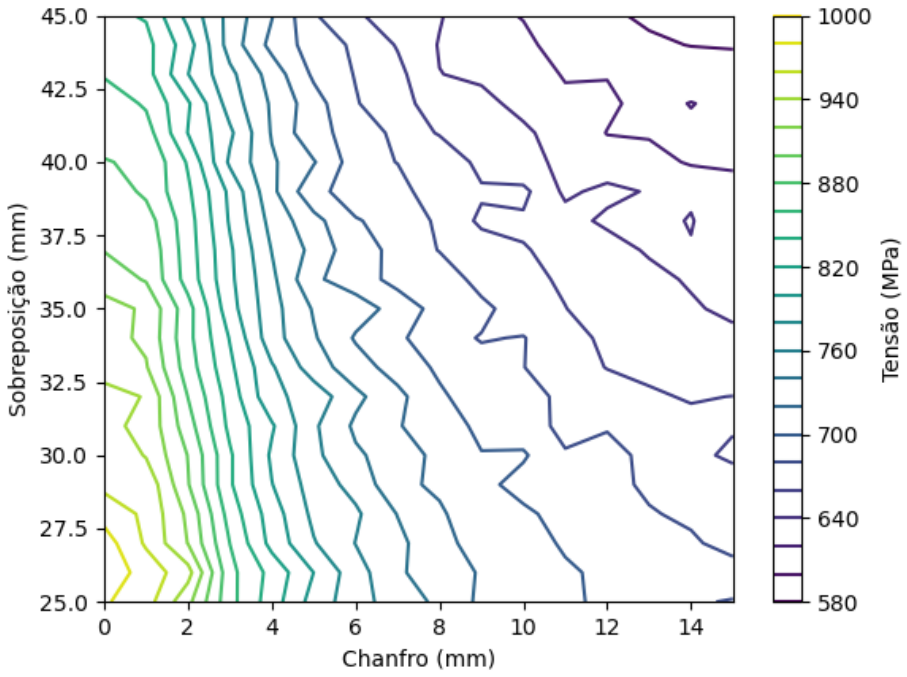


Fonte: o autor

Portanto, os métodos exatos utilizados neste trabalho são: Nelder-Mead, Powell, COBYLA e L-BFGS-B, implementados no submódulo optimize.minimize da SciPy, uma biblioteca de programação científica open source em linguagem python.

Outro fator que deve ser levado em consideração são os múltiplos mínimos locais (ruídos) encontrados neste tipo de função, fazendo com que os algoritmos de otimização local selecionados possam ficar estagnados em mínimos locais. Métodos de otimização global, como os Algoritmos Genéticos (seção 2.2.5), por exemplo, têm a capacidade de realizar uma avaliação mais abrangente do espaço de busca, escapando assim de ótimos locais. A Figura 16 apresenta, por meio de um gráfico de curvas de nível, o espaço de busca formado pela variação da largura da sobreposição e do tamanho do chanfro em uma junta adesiva.

Figura 16 – Função objetivo: Tensão máxima de delaminação



Fonte: o autor

Portanto, a escolha do AG como método aproximado, se deve ao fato de ele ser livre de derivadas e de não ter seu desempenho afetado por descontinuidades da função ou na sua derivada. Além disso, por utilizar operadores estatísticos, consegue explorar de forma mais eficiente o espaço de busca. O AG utilizado foi implementado pelo submódulo pymoo.optimize da Pymoo, uma biblioteca open source de otimização multiobjetivo em linguagem python.

4.2 Testes dos métodos usando funções de estudo

Uma vez definidos os métodos de otimização com potencial para resolver o problema de otimização estrutural, é necessário fazer a parametrização adequada das funções das bibliotecas SciPy e Pymoo para melhorar a precisão do resultado e/ou diminuir o tempo de execução.

Ao invés de avaliar os métodos de otimização diretamente no problema *benchmark*, cuja função objetivo é ruidosa, foi feita uma avaliação preliminar em funções *benchmark* da literatura, a fim de compreender o comportamento dos métodos selecionados. Estas funções, aqui chamadas de “Funções de estudo”, possuem características distintas comuns em problemas de otimização:

1. **Função Rosenbrock (Figura 17a):**

$$f(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i + 1)^2] \tag{4.1}$$

onde d é igual ao número de dimensões e $x_i \in [-2.048, 2.048]$ para todo $i = 1, ..., d$ e seu mínimo global é $f(x_i) = 0$ quando $x_i = 1$. A função Rosenbrock é unimodal (possui apenas um ponto de extremo) e o mínimo global encontra-se num vale estreito e parabólico. Contudo, embora este vale seja fácil de encontrar, a convergência para o mínimo é difícil (SURJANOVIC; BINGHAM, 2015).

2. **Função Sphere (Figura 17b):**

$$f(x) = \sum_{i=1}^{d} x_i^2 \tag{4.2}$$

onde d é igual ao número de dimensões e $x_i \in [-5.12, 5.12]$ para todo $i = 1, ..., d$ e seu mínimo global é $f(x_i) = 0$ quando $x_i = 0$. A função Sphere possui d mínimos locais, exceto o global. É contínua e unimodal (SURJANOVIC; BINGHAM, 2015).

3. **Função Ackley (Figura 17c):**

$$f(\mathbf{x}) = -a\ \exp\left(-b\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(c\ x_i)\right) + a + \exp(1) \tag{4.3}$$

onde d é igual ao número de dimensões e $x_i \in [-32.768, 32.768]$ para todo $i = 1, ..., d$ e seu mínimo global é $f(x_i) = 0$ quando $x_i = 0$. A função Ackley é multimodal (possui mais de um ponto de extremo). Esta função representa um risco para algoritmos de otimização, particularmente os algoritmos de escalada, pois podem ficar presos em um de seus muitos mínimos locais. Os valores de variáveis recomendados usados para a função Ackley são $a = 20$, $b = 0.2$ e $c = 2\pi$ (SURJANOVIC; BINGHAM, 2015).

A Figura 18 apresenta um resumo das características das funções de estudo selecionadas.

Figura 18 – Resumo das funções de estudo selecionadas

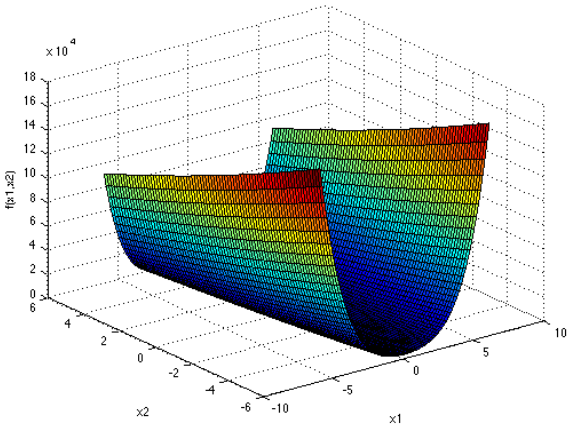
Função	Valor Mínimo	Mínimos Globais	Mínimos Locais
Rosenbrock	0	Infinitos	0
Sphere	0	1	0
Ackley	0	1	Muitos

Fonte: o autor

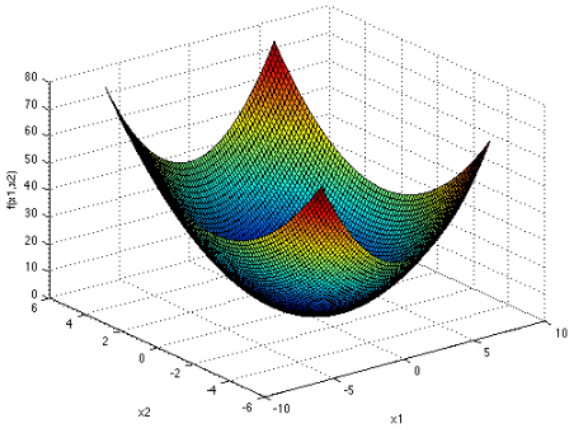
Para os métodos aproximados, as funções de estudo foram implementadas utilizando a biblioteca Pymoo, pelo submódulo pymoo.problems. Para os métodos exatos, duas bibliotecas distintas foram utilizadas:

1. A função Ackley foi implementada pela benchmark-functions, uma biblioteca open source com uma coleção de funções de benchmark escrita em python, adequada para avaliar o desempenho de problemas de otimização com métodos exatos.

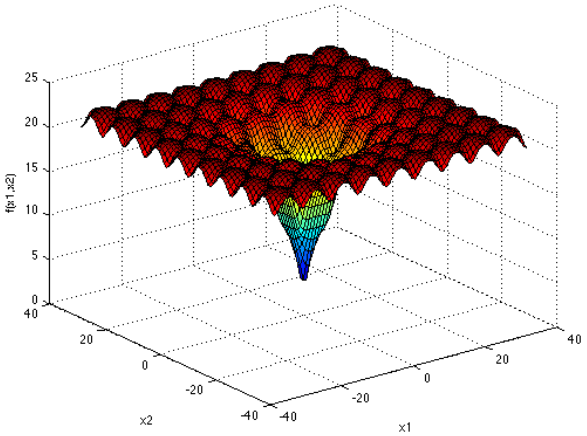
Figura 17 – Gráfico em 2 dimensões das funções de estudo



(a) Rosenbrock



(b) Sphere



(c) Ackley

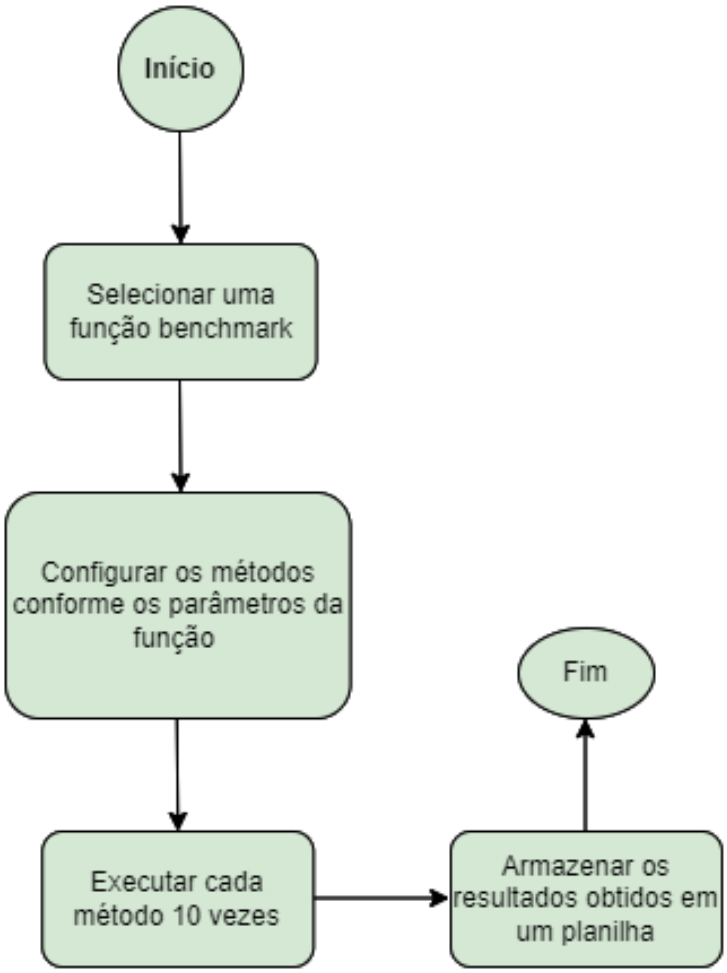
Fonte: Surjanovic e Bingham (2015)

2. As funções Sphere e Rosenbrock foram implementadas pela OptimizationTestFunctions, uma biblioteca open source em python com uma coleção de funções de teste de otimização e alguns métodos úteis para trabalhar com elas.

A biblioteca SciPy, ao implementar os métodos irrestritos, disponibiliza o recurso de restrições de caixa, possibilitando a definição de limites inferiores e superiores para cada variável de decisão. Portanto, os métodos Nelder-Mead, Powell e L-BFGS-B, apesar de irrestritos, terão restrições de caixa. O método COBYLA, apesar de ser restrito, terá suas restrições criadas de modo a funcionarem como restrições de caixa.

Para avaliar e comparar os algoritmos exatos, cada método foi executado 10 vezes e o tempo de execução de cada algoritmo foi calculado pela diferença entre o momento de início e fim da execução, cronometrados pela função default_timer(), que retorna o valor (em segundos) do momento atual, do módulo *timeit* do python. Esse módulo é uma ferramenta útil para programadores python quando existe a necessidade de verificar o tempo de execução de algum trecho de código. A Figura 19 mostra o fluxograma da etapa de testes com as funções de estudo.

Figura 19 – Fluxograma da execução dos testes com as funções de estudo



Fonte: o autor

Todos os métodos exatos utilizaram os parâmetros padrão e o mesmo número de dimensões e x_0: 50 dimensões e $x_0 = [1, 1, ..., 1, 1]$.

Para avaliar o algoritmo aproximado, o AG também foi executado 10 vezes (Figura 19) e o tempo de execução também foi cronometrado pela função default_timer().

A Figura 20 apresenta a parametrização utilizada em cada função:

Figura 20 – Parametrização do método aproximado

Função	Pop	Gen	Amostragem	Crossover	Mutação	Outros
Rosenbrok	100	150	BinaryRandomSampling	TwoPointCrossover	BitflipMutation	eliminate_duplicates
Sphere	100	100	Padrão	Padrão	Padrão	eliminate_duplicates
Ackley	100	200	Padrão	Padrão	Padrão	eliminate_duplicates

Fonte: o autor

Na Figura 20:

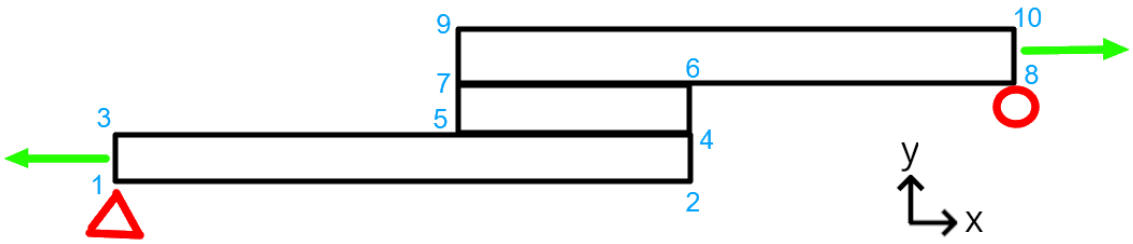
- a coluna “Pop” corresponde ao número de indivíduos de cada geração;
- a coluna “Gen” corresponde ao número de vezes que o método irá executar;
- a coluna “Amostragem” corresponde ao operador de amostragem utilizado no método;
- a coluna “Crossover” corresponde ao operador de crossover utilizado no método;
- a coluna “Mutação” corresponde ao operador de mutação utilizado no método;
- a coluna “Outros” corresponde qualquer outra parametrização realizada;

4.3 Otimização geométrica do problema *benchmark*

Como abordado na seção 2.4, no presente trabalho a função objetivo analisada será dada pela máxima tensão de delaminação, que é a componente da tensão na direção Y. Sendo assim, seguindo a metodologia estabelecida, uma vez estudado e avaliado o comportamento dos métodos para funções analíticas, foi realizada a avaliação para a função objetivo numérica, que neste trabalho é estimada a partir de um modelo de elementos finitos.

A Figura 21 apresenta um esquema da geometria da junta adesiva, a partir da qual é gerada a função objetivo (problema *benchmark*). Nesta figura são indicadas as condições de contorno: aplicação da carga (verde) e restrição dos graus de liberdade (vermelho), e a numeração dos *key points* utilizados para a modelagem no pyAnsys. É importante observar que a figura faz uma representação meramente ilustrativa da geometria, uma vez que apresenta, por exemplo, a espessura do adesivo de maneira desproporcional. Na geometria real, o aderente possui 1.6 milímetros de espessura, 100.0 milímetros de comprimento e 25.0 milímetros de largura. O adesivo possui 0.2 milímetros de espessura. Estes valores podem ser vistos na Tabela 2.

Figura 21 – Esquema da junta adesiva modelada para o pyAnsys



Fonte: o autor

Tabela 2 – Parâmetros constantes do problema

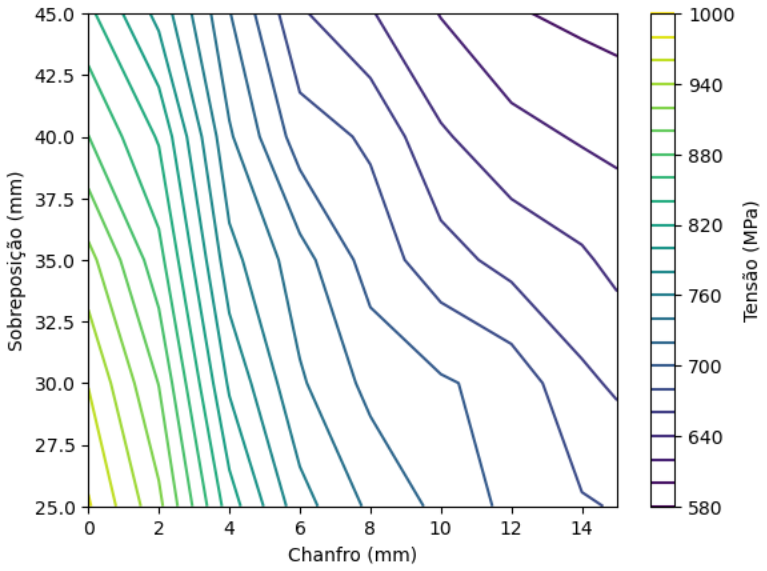
Nome	**Significado**	**Valor (mm)**
T_ADV	Espessura do adesivo	0.2
T_ADH	Espessura do aderente	1.6
L_ADH	Comprimento do aderente	100

WIDTH	Largura do aderente	25

Fonte: o autor

Neste modelo, busca-se melhorar a distribuição das tenções através da criação de chanfros externos, para isso é necessário variar no eixo x os *key points* 2 e 9, que se movem em direções opostas, na mesma medida. Além disso, pode-se alterar também a largura da sobreposição, para isso é necessário variar no eixo x os *key points* 5, 7, 8, 9 e 10 que se movem na mesma direção, na mesma medida. Além disso, algumas restrições foram impostas ao modelo: o chanfro não pode ultrapassar 15.0 milímetros e a sobreposição deve variar entre 25 e 45 milímetros. A Figura 22 apresenta, por meio de um gráfico de curvas de nível, o espaço de busca referente à máxima tensão de delaminação, formado pela variação da largura da sobreposição e do tamanho do chanfro.

Figura 22 – Função objetivo: Tensão máxima de delaminação



Fonte: o autor

Para variar os chanfros externos e o tamanho da sobreposição, duas variáveis de decisão serão utilizadas: `chanfro` (valor de deslocamento dos *key points* 2 e 9) e `sobreposicao` (valor de deslocamento dos *key points* 5, 7, 8, 9 e 10).

Assim, a configuração de cada *key point* do modelo (Figura 21) pode ser visto na Tabela 3. As constantes são apresentadas na Tabela 2.

Tabela 3 – *Key points* para a definição da geometria no pyAnsys

Key point	**Coordenada X**	**Coordenada Y**
1	0	0
2	L_ADH - `chanfro`	0

3	T_ADH	0
4	L_ADH	T_ADH
5	L_ADH - `sobreposicao`	T_ADH
6	L_ADH	T_ADH + T_ADV
7	L_ADH - `sobreposicao`	T_ADH + T_ADV
8	2*L_ADH - `sobreposicao`	T_ADH + T_ADV
9	L_ADH - `sobreposicao` + `chanfro`	2*T_ADH + T_ADV
10	2*L_ADH - `sobreposicao`	2*T_ADH + T_ADV

Fonte: o autor

Nos *key points* 8 e 10 o eixo x é decrementado pela variável `sobreposicao`, isso se dá pelo fato dos *key points* 5 e 7 "caminharem" em direção à origem conforme o aumento da sobreposição. Para não comprometer as proporções do modelo original, todo o aderente superior também "caminha" para à origem na mesma medida.

Conforme apresentado na Seção 2.4.1, o problema de otimização para minimizar tensão de delaminação no adesivo (tensão na direção Y) pode ser definido da seguinte forma:

Minimizar

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl} \qquad i,j = y \tag{4.4}$$

Sujeito a:

$$chanfro \geq 0 \tag{4.5}$$

$$chanfro \leq 15 \tag{4.6}$$

$$sobreposicao \geq 25 \tag{4.7}$$

$$sobreposicao \leq 45 \tag{4.8}$$

Na equação 4.4, σ é o tensor de tensões (ou tensor de Cauchy), C é o tensor constitutivo de 4ª ordem e ε é o tensor de deformações (SCHON, 2013). As condições de contorno são as seguintes (Conforme a Figura 21):

- Carga aplicada: $5kN$;
- *Key point* 1: Apoio fixo;
- *Key point* 8: Apoio móvel (Movimentação apenas na direção x).

Foi considerado um aderente de alumínio e um adesivo epóxi de rigidez intermediária. As propriedades destes materiais utilizadas para gerar o tensor C são as seguintes:

- Módulo de elasticidade do Aderente: $70GPa$

- Coeficiente de Poisson do Aderente: $0,33$
- Módulo de elasticidade do Adesivo: $2522MPa$
- Coeficiente de Poisson do Adesivo: $0,4$

4.4 Parametrização dos métodos no problema benchmark

Para resolver este problema *benchmark*, foi realizado um levantamento dos parâmetros utilizados na implementação de cada método exato pela biblioteca SciPy.

Inicialmente buscou-se levantar os parâmetros gerais, que são aqueles que a maioria dos métodos utiliza e que não interferem diretamente no seu desempenho. As Tabelas 4 e 5 apresentam, respectivamente, a descrição dos parâmetros gerais e os valores utilizados em cada método.

Tabela 4 – Parâmetros gerais

Parâmetro	**Definição**
fun	função objetivo a ser minimizada
x0	chute inicial
method	método solucionador
bounds	limites das variáveis de decisão
maxfev	número máximo de avaliações de função
tol	precisão final na otimização (não garantida precisamente)

Fonte: o autor

Tabela 5 – Valores utilizados para parâmetros gerais

Parâmetro	**Powell**	**Nelder-Mead**	**COBYLA**	**L-BFGS-B**
fun	equação 4.4	equação 4.4	equação 4.4	equação 4.4
x0	[10,30]	[10,30]	[10,30]	[10,30]
method	"Powell"	"Nelder-Mead"	"COBYLA"	"L-BFGS-B"
bounds	[(0, 15), (25, 45)]	[(0, 15), (25, 45)]		[(0, 15), (25, 45)]
maxfev	1e+20	1e+5		
tol				1e-10

Fonte: o autor

Em seguida foram levantados na Tabela 6, os parâmetros que apenas um único método utiliza ou que influenciam diretamente o desempenho da busca.

Tabela 6 – Parâmetros específicos

Powell

Parâmetro	**Descrição**
direc	conjunto inicial de vetores de direção
xtol	erro relativo na solução aceitável para convergência

Nelder-Mead

Parâmetro	**Descrição**
initial_simplex	simplex inicial
xatol	erro absoluto entre iterações, aceitável para convergência

COBYLA

Parâmetro	**Descrição**
constraints	lista de restrições
catol	tolerância absoluta para violações de restrições

L-BFGS-B

Parâmetro	**Descrição**
maxls	número máximo de etapas de busca de linha (por iteração)

Fonte: o autor

Para compreender melhor o funcionamento dos métodos sob a influência de seus parâmetros específicos, foram realizados testes de execução com variações desses parâmetros. Para os métodos Powell, Nelder-Mead e COBYLA foram feitas 4 rodadas de testes: a cada rodada o valor do parâmetro específico foi alterado, os demais parâmetros foram fixados (conforme a Tabela 5) e o método foi executado 5 vezes. As Tabelas 7, 8 e 9 apresentam os parâmetros específicos variados em cada rodada para cada método.

Tabela 7 – Testes de parâmetros do método Powell

Parâmetro	**Rodada 1**	**Rodada 2**	**Rodada 3**	**Rodada 4**
direc	[1,0],[0,1]	[1,0],[0,1]	[0,2],[2,0]	none

Fonte: o autor

Tabela 8 – Testes de parâmetros do método Nelder-Mead

Parâmetro	**Rodada 1**	**Rodada 2**	**Rodada 3**	**Rodada 4**
initial_simplex	[1,0],[0,1],[-1,0]	[-1,0],[1,1],[0,0]	[20,10],[15,20],[10,30]	none

Fonte: o autor

Tabela 9 – Testes de parâmetros do método COBYLA

Parâmetro	**Rodada 1**	**Rodada 2**	**Rodada 3**	**Rodada 4**
tol	1e-2	1e-5	1e-10	none
catol	1e-2	1e-5	1e-10	none

Fonte: o autor

Na implementação do método L-BFGS-B da biblioteca SciPy, o único parâmetro específico disponível é o maxls, que funciona apenas como limitador máximo do número de execuções e não interfere na convergência do método. Portanto, não foram realizados testes de variação com este parâmetro.

Os métodos aproximados não são o foco desta pesquisa e a implementação destes métodos pela biblioteca Pymoo não foi explorada, então também não foram realizados testes de variação para o AG. A Tabela 10 apresenta uma descrição dos parâmetros utilizados.

Tabela 10 – Descrição dos parâmetros do AG

Parâmetro	**Descrição**
pop_size	tamanho da população
n_gen	número de gerações
eliminate_duplicates	elimina duplicatas após cruzar a população dos pais e dos descendentes

Fonte: o autor

Nessas condições, os métodos AG e L-BFGS-B foram aplicados diretamente no problema benchmark, sem avaliação individual de parâmetros. Foram realizadas 5 execuções de cada método. As Tabelas 11 e 12 apresentam a parametrização utilizada em cada método.

Tabela 11 – Parametrização do método L-BFGS-B

Parâmetro	**Exec. 1**	**Exec. 2**	**Exec. 3**	**Exec. 4**	**Exec. 5**
maxls	1000	1000	1000	1000	1000

Fonte: o autor

Tabela 12 – Parametrização do método AG

Parâmetro	**Exec. 1**	**Exec. 2**	**Exec. 3**	**Exec. 4**	**Exec. 5**
pop_size	50	50	50	50	50
n_gen	10	10	10	10	10
eliminate_duplicates	True	True	True	True	True

Fonte: o autor

5 Resultados

Neste capítulo, serão analisados os resultados dos testes dos métodos quando executados para as funções de estudo apresentadas na seção 4.2, os resultados dos testes de variação de parâmetros descritos na seção 4.4 e os resultados da otimização geométrica do problema benchmark da seção 4.3.

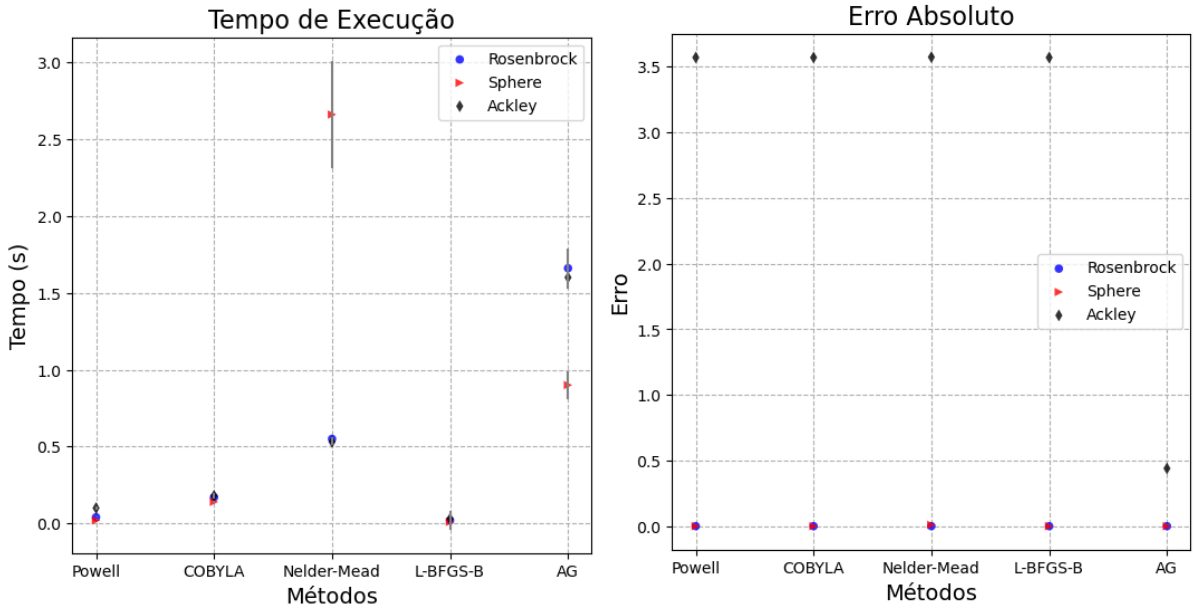
5.1 Testes com funções de estudo

Nesta seção, serão analisados os tempos e erros de execução de cada método durante a otimização de cada função de estudo. Estes dados serão utilizados como medida para uma comparação preliminar entre os métodos.

Como mencionando na seção 4.2, cada algoritmo foi executado 10 vezes para cada uma das 3 funções de estudo. Ao fim de cada execução os resultados foram armazenados em tabelas disponibilizadas no Apêndice A.

A Figura 23a apresenta a média dos tempos de execução e a Figura 23b apresenta os erros absolutos, calculados pela equação 2.6, para cada método.

Figura 23 – Resultados dos teste com as funções de estudo



(a) Média do tempo de execução

(b) Erro absoluto

Fonte: o autor

É importante ressaltar que os métodos exatos utilizados também são determinísticos, isto é, eles produzem o mesmo resultado todas as vezes que são executados com as mesmas entradas. Por exemplo, as 10 execuções do COBYLA, produziram os mesmos resultados. Em

contrapartida, o método aproximado utilizado também é estocástico, isto é, ele incorpora aleatoriedade em sua execução, o que significa que o resultado pode variar de uma execução para outra, mesmo com as mesmas entradas. Portanto, as 10 execuções do AG não produziram os mesmos resultados, sendo selecionado para comparação o melhor resultado.

5.1.1 **Comparação entre métodos exatos**

Para as funções Rosenbrock e Sphere, os métodos exatos foram precisos ao convergirem para o valor mínimo, obtendo um erro absoluto igual a 0 (Figura 23b). Estes valores são aceitáveis dentro do contexto apresentado, pois ambas as funções não são complexas, além de serem contínuas e diferenciáveis. A Tabela 13 apresenta as médias de tempo para as funções Rosenbrock e Sphere.

Tabela 13 – Tempos médios para funções Rosenbrock e Sphere

Método	**Rosenbrok**	**Sphere**
Powell	0,04s	0,02s
COBYLA	0,17s	0,14s
L-BFGS-B	0,02s	0,01s
Nelder-Mead	0,55s	2,56s

Fonte: o autor

Analisando esses valores, é importante fazer a seguinte observação: o método L-BFGS-B, por ser baseado em gradiente, ao otimizar funções simples, contínuas e diferenciáveis, é mais rápido que os métodos livres de derivada. Em contrapartida, o método de Powell, utiliza direções conjugadas para a busca, realiza menos iterações e é mais rápido que o COBYLA e o Nelder-Mead.

Ao analisar o erro absoluto dos métodos exatos para a função Ackley (Figura 23b), levando em consideração que esses métodos são de busca local, conclui-se que os resultados são aceitáveis, pois evidenciam que todos os métodos exatos selecionados ficaram ”presos“ em algum mínimo local. Isso acontece porque, diferentemente das funções Rosenbrock e Sphere, a função Ackley possui múltiplos mínimos locais e um único mínimo global, dificultando a busca eficiente pelo mínimo global, especialmente para métodos que dependem da informação local, como os métodos exatos selecionados neste trabalho.

Além disso, à medida que a dimensionalidade do espaço de busca aumenta, a função Ackley torna-se ainda mais desafiadora de otimizar. Isso ocorre porque o número de mínimos locais aumenta exponencialmente com o número de variáveis, tornando a busca pelo mínimo global ainda mais complexa, a depender do chute.

5.1.2 **Comparação entre métodos exatos e AG**

Todos os métodos exatos utilizados neste trabalho foram selecionados por realizarem busca direta. Essa escolha foi baseada na função objetivo trabalhada, caracterizada por ser

ruidosa. Com exceção do L-BFGS-B, os demais métodos (Powell, Nelder-Mead e COBYLA) são apropriados para lidar com ruídos, pois não dependem do cálculo do gradiente.

Analisando o desempenho dos métodos exatos selecionados na função Ackley, a escolha de um método não determinístico (aproximado), neste caso o AG, se torna clara: métodos aproximados, dada sua natureza probabilística, conseguem explorar espaços de busca mais vastos e diversificados, além de não ficarem presos em mínimos locais. E, apesar de não haver garantia de encontrar o mínimo global, esses métodos conseguem se aproximar do mesmo, o que pode ser visto no resultado do AG para a função Ackley.

Além disso, levando em consideração que a função objetivo poderá ser desafiadora para os métodos exatos de busca local, o AG novamente se torna uma escolha adequada.

5.2 Testes de variação de parâmetros no problema benchmark

Nesta seção, serão analisados os tempos e resultados de execução de cada método durante as rodadas de testes de variação de parâmetros. Por meio dessa análise é possível avaliar qual variação de parâmetro produziu os melhores resultados.

Como dito na seção 4.4, cada método foi submetido a 4 rodadas de teste e em cada rodada o método foi executado 5 vezes. A cada rodada o valor do parâmetro específico foi alterado e os demais parâmetros ficaram fixos. As tabelas com todos os dados dos testes podem ser encontradas no Apêndice B.

5.2.1 **Powell**

Para o método Powell, o parâmetro variado foi o *direc*: conjunto inicial de vetores de direção. A Tabela 14 apresenta, para cada rodada, o valor do parâmetro, a média do resultado (máxima tensão na direção Y) e a média do tempo de execução em segundos.

Tabela 14 – Resultados da variação de parâmetros do método Powell

Rodada	**Valor**	**Resultado médio (MPa)**	**Tempo médio (s)**
1	[0,1],[1,0]	613,01	728,60
2	[1,0],[0,1]	611,23	687,45
3	[0,2],[2,0]	612,31	694,91
4	none	602,46	829,76

Fonte: o autor

Na Tabela 14 é possível observar que o menor resultado e o menor tempo foram obtidos nas rodadas 4 e 2, respectivamente. Isso implica que, para este problema, o parâmetro *direc* foi mais relevante para a diminuição do tempo de execução do que para a convergência para o mínimo.

A Tabela 15 apresenta, como exemplo, um detalhamento da rodada 3. Analisando a coluna “F(x)” nota-se que os valores variam a cada execução, mesmo recebendo os mesmo

parâmetros. Isso ocorre devido a natureza da função objetivo (equação 4.4), pois dado seu caráter numérico, para cada nova avaliação da função objetivo, uma nova malha e uma nova solução para o modelo de elementos finitos são geradas. Aliado a isso, como o algoritmo Powell analisa apenas os dois pontos anteriores, com os ruídos da função e acumulo de erros de arredondamento, perde-se precisão.

Tabela 15 – Rodada 3 do método Powell

Rodada	**F(x) (MPa)**	**Tempo (s)**
1	612,24	655,26
2	617,67	688,80
3	612,75	685,67
4	609,35	703,18
5	609,55	741,63
Média	612,31	694,91
Desvio Padrão	3,36	31,41

Fonte: o autor

5.2.2 **Nelder-Mead**

No método Nelder-Mead, o parâmetro variado foi o *initial_simplex*: simplex inicial (os três pontos iniciais que vão delimitar a primeira área de busca). A Tabela 16 apresenta, para cada rodada, o valor do parâmetro, a média do resultado (máxima tensão na direção Y) e a média do tempo de execução em segundos.

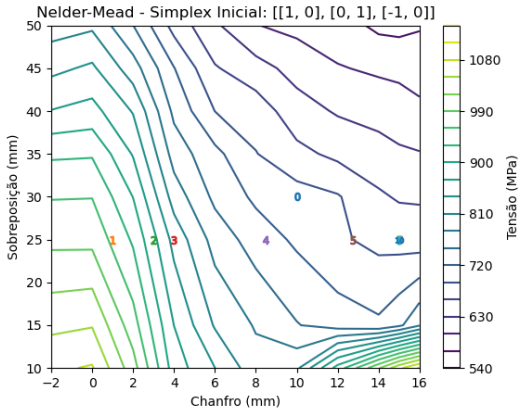
Tabela 16 – Resultados da variação de parâmetros do método Nelder-Mead

Rodada	**Valor**	**Resultado médio (MPa)**	**Tempo médio (s)**
1	[1,0],[0,1],[-1,0]	678,64	139,74
2	[2,2],[7,7],[10,10]	678,64	75,17
3	[20,10],[15,20],[10,30]	592,08	178,02
4	none	592,08	299,35

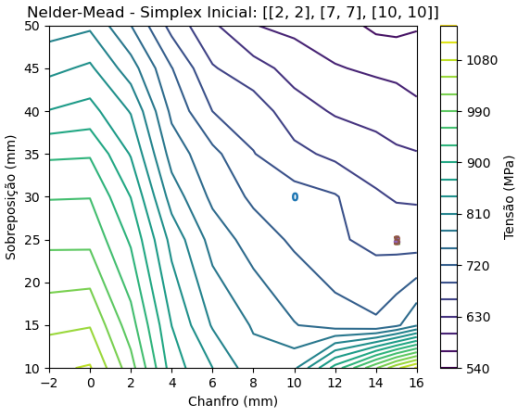
Fonte: o autor

Na Tabela 16 é possível observar que a medida que o simplex inicial foi aumentando, o tempo de execução foi diminuindo. Isso implica que, para este problema, quanto maior a primeira área de busca, menos tempo será gasto, pois o tamanho dessa área de busca influencia diretamente na convergência do algoritmo. Na Figura 24 observa-se que na maior área de busca (Figura 24c) o algoritmo "caminha" para o mínimo global, enquanto as menores áreas de busca (Figuras 24a e 24b) não é possível ver o algoritmo se aproximar do mínimo global. Observa-se ainda que, mesmo com ponto $(20, 10)$ fora da região viável, já que o maior chanfro permitido é $15mm$, o algoritmo foi convergente na rodada 3.

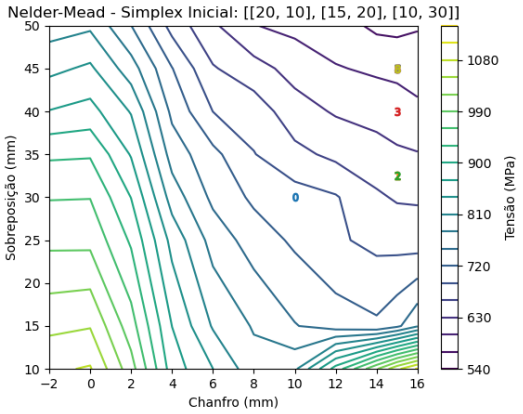
Figura 24 – Rodadas do Nelder-Mead



(a) Rodada 1



(b) Rodada 2



(c) Rodada 3

A Tabela 17 apresenta, como exemplo, um detalhamento da rodada 3. Analisando a coluna “F(x)”, nota-se que, ao receberem os mesmos parâmetros, os valores não variam a cada execução. Este comportamento é o contrário do método Powell. Isso acontece porque o método Powell, na presença de ruído, precisa realizar muitas avaliações e/ou não converge, se comportando de modo aleatório (PLAGIANAKOS; VRAHATIS, 2001). Além disso, o algoritmo Nelder-Mead avalia quatro pontos (três iniciais e o próximo), em cada iteração, o que contribui para sua maior robustez.

Tabela 17 – Rodada 3 do método Nelder-Mead

Rodada	**F(x) (MPa)**	**Tempo (s)**
1	592,08	181,74
2	592,08	168,49
3	592,08	184,56
4	592,08	185,36
5	592,08	169,96
Média	592,08	178,02

Desvio Padrão	0	8,16

Fonte: o autor

5.2.3 **COBYLA**

Para o método COBYLA, dois parâmetros foram variados: *tol* (precisão final na otimização) e *catol* (tolerância absoluta para violações de restrições). A Tabela 18 apresenta, para cada rodada, o valor dos parâmetros, a média do resultado (máxima tensão na direção Y) e a média do tempo de execução em segundos.

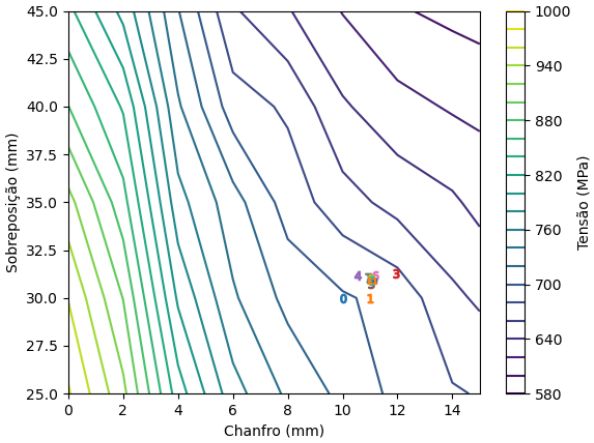
Tabela 18 – Resultados da variação de parâmetros do método COBYLA

Rodada	**Valor**	**Resultado médio (MPa)**	**Tempo médio (s)**
1	1e-2	664,68	265,33
2	1e-5	667,85	512,17
3	1e-10	674,42	775,16
4	none	672,64	353,19

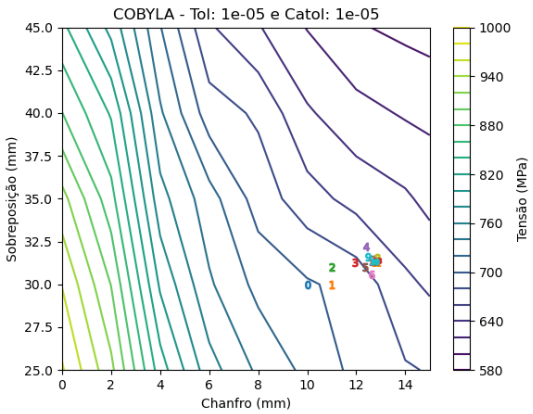
Fonte: o autor

Na Tabela 18 é possível observar que a medida que a tolerância foi diminuindo, o tempo de execução aumentou. Conclui-se que para este problema, quanto maior a tolerância, menos tempo será gasto e menor será o resultado. Na Figura 25, nota-se que quanto menor a tolerância, mais interações são realizadas, mas sem melhorias significativas na exploração do espaço de busca.

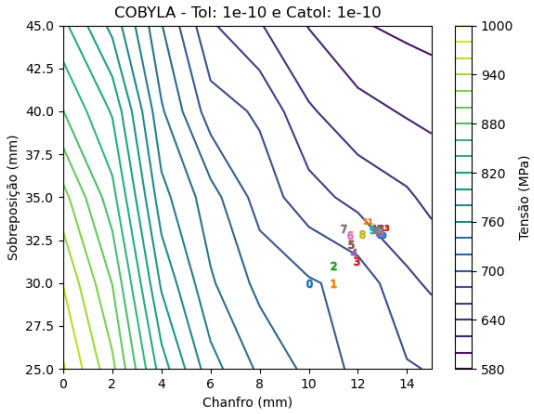
Figura 25 – Rodadas do COBYLA



(a) Rodada 1



(b) Rodada 2



(c) Rodada 3

A Tabela 19 apresenta, como exemplo, um detalhamento da rodada 3 do método COBYLA. Analisando a coluna “F(x)” nota-se que os valores variam a cada execução, mesmo recebendo os mesmo parâmetros. Isso possivelmente ocorre porque o método, a cada iteração, faz uma linearização do problema, o que envolve muitos arredondamentos, que vão atuar como ruídos adicionais numa função objetivo que já é ruidosa. A nova direção de busca é calculada com base no resultado do PPL e qualquer erro pode variar esta direção.

Tabela 19 – Rodada 3 do método COBYLA

Rodada	**F(x) (MPa)**	**Tempo (s)**
1	669,93	753,24
2	676,60	817,82
3	673,99	829,84
4	670,00	738,75
5	681,60	736,15
Média	674,42	775,16
Desvio Padrão	4,91	45,11

Fonte: o autor

5.2.4 **Análise geral**

A Tabela 20 apresenta as rodadas de cada método que obtiveram o menor resultado médio ao otimizar o problema benchmark, juntamente com o tempo de execução médio em segundos e o valor do parâmetro específico utilizado.

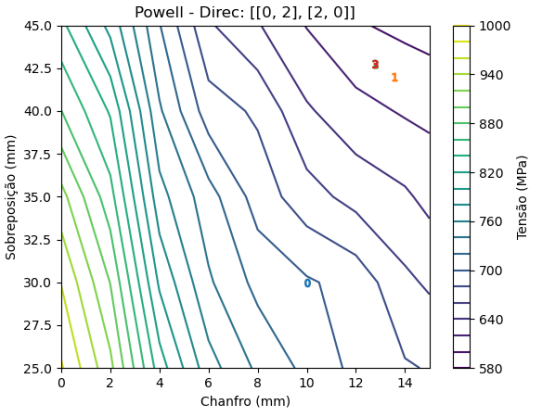
Tabela 20 – Menor resultado médio na variação de parâmetros

Método	**Resultado médio (MPa)**	**Tempo médio (s)**	**Parametrização**
Powell	602,46	829,76	direc = none
Nelder-Mead	592,08	299,35	initial_simplex = none
COBYLA	664,68	265,33	tol = catol = 1e-2

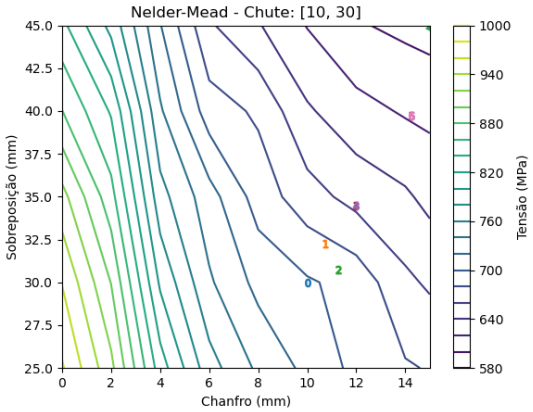
Fonte: o autor

Analisando a Figura 26 e os valores da Tabela 20, observa-se que o método COBYLA (Figura 26c) realizou o maior número de iterações devido à alta precisão utilizada, o método Powell (Figura 26a) obteve o menor número de iterações e chegou próximo ao mínimo global e o método Nelder-Mead (Figura 26b), sem um simplex inicial, foi progressivamente “caminhando” até encontrar o mínimo global.

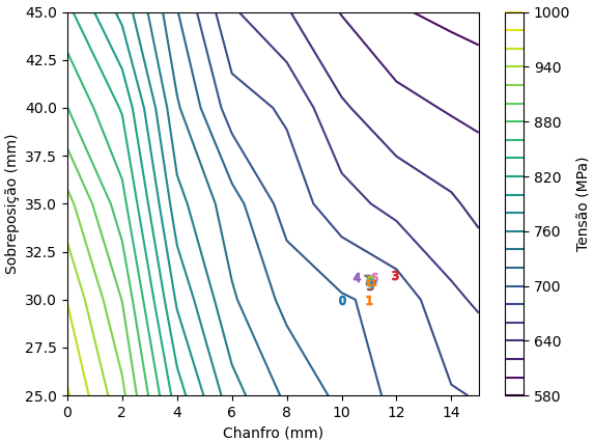
Figura 26 – Menores resultados médios



(a) Powell



(b) Nelder-Mead



(c) Cobyla

5.3 Otimização geométrica do problema benchmark

Nesta seção, serão analisados os resultados da execução durante a otimização geométrica do problema benchmark. Para os métodos Nelder-Mead, Powell e COBYLA, uma das rodadas de execução foi selecionada. Para o AG e L-BFGS-B, que não foram submetidos a testes de parametrização, a rodada única de cada um foi selecionada. Estes dados serão utilizados para avaliar o desempenho de cada método para o problema de otimização estrutural.

A Tabela 21 apresenta os resultados obtidos de cada método ao minimizar função objetivo dada pela Equação 4.4. Os dados das colunas “Tensão em Y”, “Chanfro”, “Sobreposição” e “Tempo”, correspondem à média dos valores de cada execução da rodada.

Tabela 21 – Resultados

Método	**Tensão em Y**	**Chanfro**	**Sobreposição**	**Tempo**
Powell	612,31 MPa	12,10 mm	42,66 mm	11,58 min
COBYLA	674,42 MPa	11,68 mm	31,65 mm	12,92 min
Nelder-Mead	592,08 MPa	15 mm	45 mm	2,97 min

L-BFGS-B	687,78 MPa	10 mm	30 mm	14,64 min
AG	662,18 MPa	12,57 mm	33,72 mm	2,50 min

Fonte: o autor

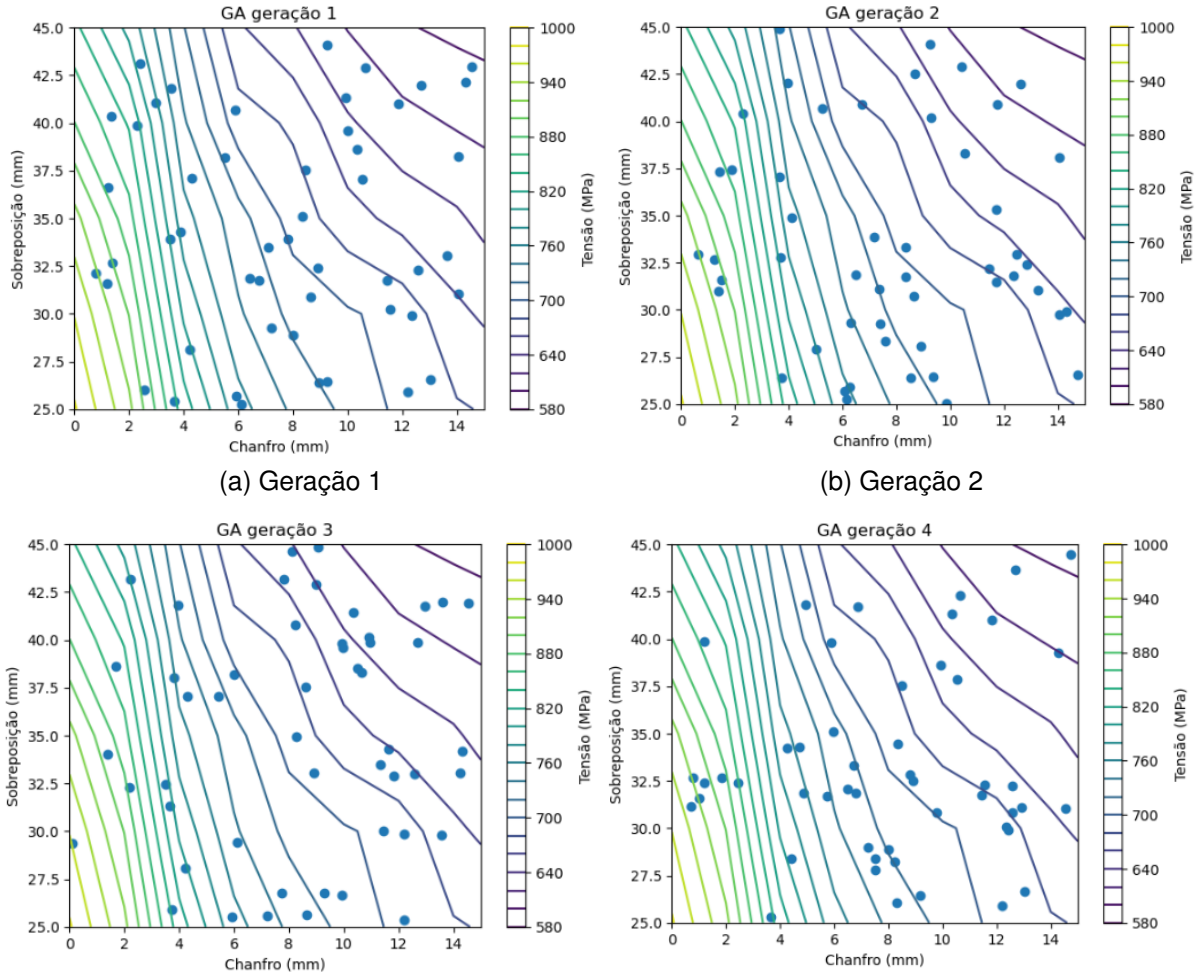
Na Seção 4.3, o problema de otimização geométrica foi apresentado e descrito na forma canônica de um problema de otimização clássico. Com essas informações, cada método foi executado a fim de resolver o problema. O código-fonte dessa aplicação é apresentado no Apêndice C.

Analisando a tabela 21, o método Nelder-Mead se destaca por obter a menor tensão em Y, e atingir consequentemente o melhor resultado, uma vez que o problema benchmark atinge o mínimo global com um chanfro de $15mm$ e uma sobreposição de $45mm$, como pode ser observado nas Figuras 16 e 22.

Em primeiro lugar, destaca-se o fato do AG não ter encontrado o mínimo global, isto é, a menor tensão de delaminação. Apesar deste método ser capaz de explorar mais amplamente o espaço de busca, se aproximando do mínimo global, não foi realizada uma avaliação aprofundada dos parâmetros disponíveis, sendo adotados os valores e recursos padrões da biblioteca Pymoo. Contudo, o AG obteve a terceira menor tensão, superando os métodos COBYLA e L-BFGS-B, porém com um tempo médio de execução compatível com o do método Nelder-Mead, o qual alcançou a tensão mínima.

Nas Figuras 27 e 28 pode-se comparar os “caminhos” percorridos pelos métodos AG e Nelder-Mead, respectivamente. Nota-se que o AG explorou amplamente o espaço de busca (Figura 27), enquanto o Nelder-Mead “caminhou” progressivamente para o mínimo global (Figura 28). Além disso, para cada execução, o AG realizou 500 avaliações fundamentais, enquanto o Nelder-Mead realizou, em média, 21 avaliações fundamentais.

Figura 27 – Gerações inicias do AG em uma execução



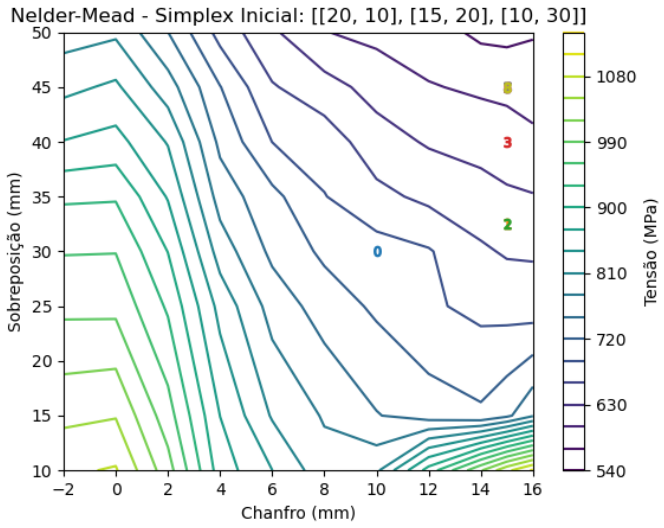
(a) Geração 1

(b) Geração 2

(c) Geração 3

(d) Geração 4

Figura 28 – Caminho do Nelder-Mead em uma execução



Fonte: o autor

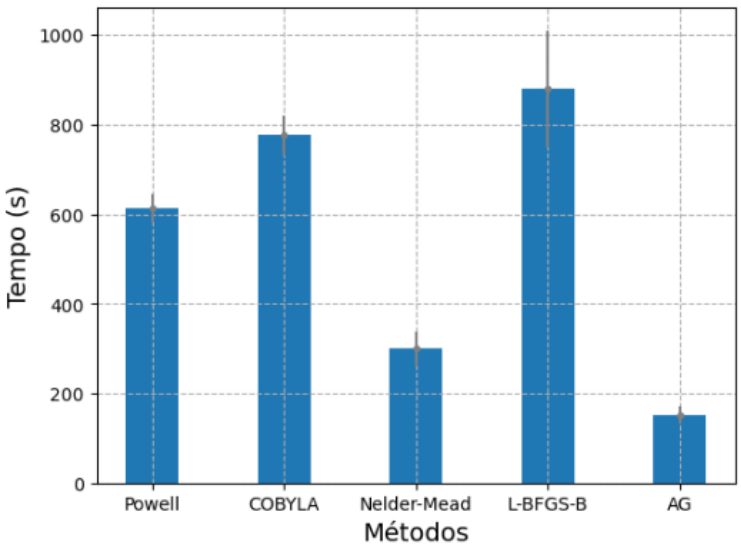
Em seguida, destaca-se o método L-BFGS-B, que apresentou a maior tensão. O exemplo do L-BFGS-B ilustra bem como é importante, em um problema de otimização, conhecermos bem as características da função objetivo de estudo. Uma vez que nossa função é caracterizada por ruídos, métodos baseados em derivadas terão um desempenho inferior aos métodos livres de derivadas, como foi explicado ao longo de todo o trabalho. Assim, o L-BFGS-B cumpre bem seu papel como contraexemplo.

Os métodos Nelder-Mead e Powell apresentaram, respectivamente, os melhores resultados. Esse fato é de grande importância, uma vez que estes métodos são desencorajados para problemas de otimização global, uma vez que tendem a ficar ”presos“ em mínimos locais. Além disso, esse resultado aponta que podemos empregar esses métodos para problemas de otimização estrutural mais simples, com garantia de bons resultados.

Também é importante ressaltar que esse resultado depende, além da natureza do problema, de uma parametrização correta. Ambos os métodos, Nelder-Mead e Powell, recebem chutes iniciais que vão ser um importante ponto de partida para a busca direta. E, uma vez que essa parametrização esteja bem feita, obtemos bons resultados.

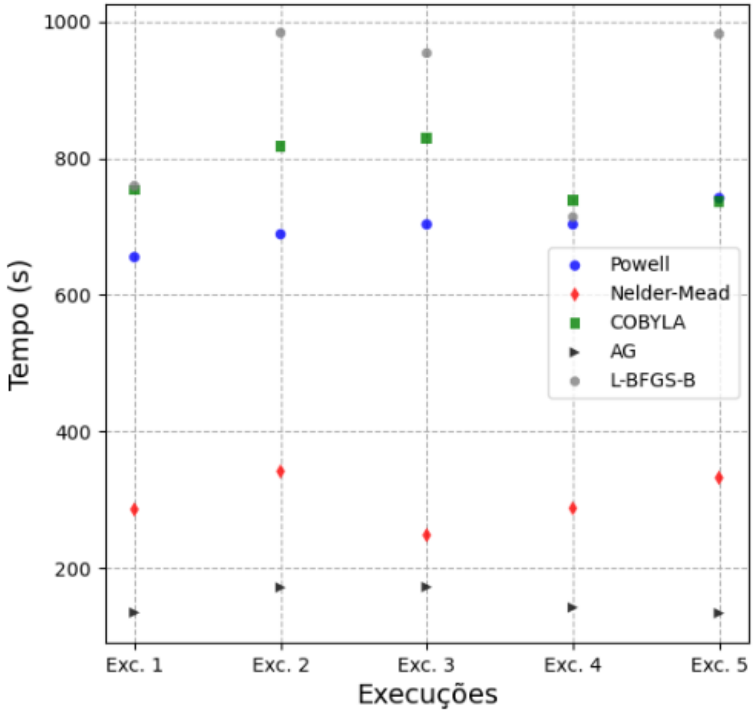
As Figuras 29 e 30 apresentam, respectivamente, o tempo médio de cada rodada e o tempo individual de cada execução. Analisando a Figura 29, observa-se que os métodos AG e Nelder-Mead foram os mais rápidos, enquanto os métodos L-BFGS-B e COBYLA foram os mais lentos. Ao mesmo tempo, nota-se uma alta variância em torno do tempo médio do L-BFGS-B, enquanto a variância dos demais métodos são menores. A Figura 30 evidencia a maior variação nos tempos de convergência do L-BFGS-B.

Figura 29 – Tempo médio por rodada



Fonte: o autor

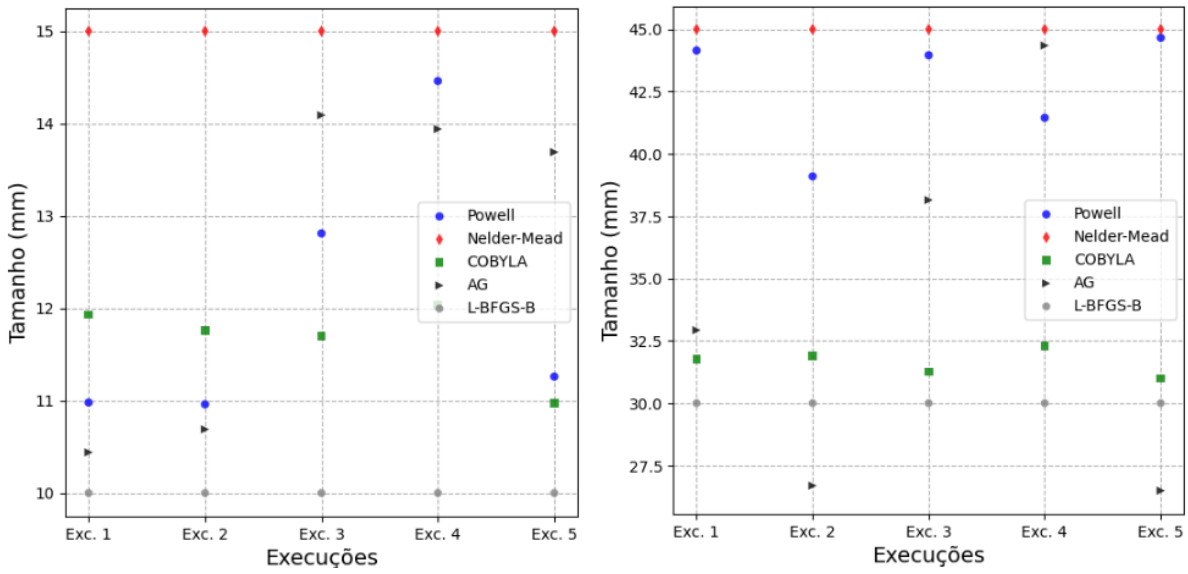
Figura 30 – Tempo de cada execução



Fonte: o autor

A Figura 31 apresenta, os valores do chanfro e sobreposição obtidos em cada execução individual. Analisando estes dados, observa-se que o Nelder-Mead e L-BFGS-B foram os métodos mais confiáveis para este problema, pois em todas as execuções os valores de chanfro e sobreposição se mantiveram iguais.

Figura 31 – Chanfro e Sobreposição



(a) Chanfro (b) Sobreposição

Fonte: o autor

A partir da análise das Figuras 30 e 31, pode-se classificar os algoritmos avaliados de acordo com as métricas de desempenho apresentadas na Seção 2.5: eficiência, confiabilidade e qualidade da resposta. No entanto, deve-se ressaltar que tal classificação é válida para os algoritmos com a parametrização investigada, tendo como finalidade a solução do problema *benchmark*. A Tabela 22 contém a classificação dos algoritmos, e a seguir são apresentados os critérios para esta classificação:

- **Eficiente (ou rápido)**: Algoritmos que na média, tiveram tempo de execução menor que 400 segundos. Este valor foi obtido da aproximação da metade do maior tempo gasto (Figura 29): o método L-BFGS-B demorou 878.52 segundos, portanto a metade é 439.26 segundos e, para se trabalhar com um valor redondo, foi realizada uma aproximação para 400 segundos. Além disso, a classificação também pode ser feita mediante uma análise de banda na Figura 29, pois os tempos estão separados em duas bandas: abaixo de 400 segundos e acima de 600 segundos.
- **Confiável (ou robusto)**: Algoritmos que na média, convergiram para o mesmo valor da função objetivo. Em termos das variáveis de decisão, considera-se robusta a solução com variância de até $1,0mm$ para o chanfro e para a sobreposição.
- **De alta qualidade**: Algoritmos que convergiram para a solução ótima, isto é, para a tupla `(chanfro, sobreposição)` = $(15, 45)$.

Tabela 22 – Classificação dos algoritmos de acordo com as métricas de desempenho

Método	**Eficiência**	**Confiabilidade**	**Qualidade**
Powell	Lento	Sensível	Questionável
Nelder-Mead	**Rápido**	**Robusto**	**Alta**
COBYLA	Lento	Sensível	Questionável
AG	**Rápido**	Sensível	Questionável
L-BFGS-B	Lento	**Robusto**	Questionável

Fonte: o autor

6 Considerações Finais

O objetivo geral deste trabalho foi a comparação dos métodos de otimização das bibliotecas SciPy e Pymoo quando aplicados ao problema de otimização geométrica de juntas adesivas.

Os métodos de busca direta da biblioteca SciPy (Nelder-Mead, Powell e COBYLA) foram avaliados em duas etapas: otimização das funções de estudo, onde nenhuma parametrização específica foi utilizada, e otimização do problema benchmark, onde cada método foi parametrizado e o impacto de cada parâmetro foi avaliado. O algoritmo genético (AG) da biblioteca Pymoo e o método exato L-BFGS-B da biblioteca SciPy foram submetidos a apenas uma etapa: otimização do problema benchmark. Todos os processos de otimização foram implementados utilizando a interface PyMAPDL.

A comparação final foi feita por meio da seleção dos resultados de cada método (exatos e aproximado) ao otimizarem o problema benchmark. Essa seleção se baseou no menor valor da máxima tensão de delaminação no adesivo (componente na direção Y).

Nos resultados selecionados, foi observado o potencial dos métodos exatos Nelder-Mead e Powell na otimização do problema benchmark, uma vez que apresentaram os melhores resultados: o Nelder-Mead demandou cerca de $3,0min$ de execução para obter $592,08MPa$, enquanto o Powell demandou cerca de $14,0min$ para obter $602,46MPa$.

A arquitetura oficial, o PyMAPDL, se comparada ao trabalho desenvolvido por Araujo (2022) no CEFET-MG, é mais simples de ser instalada, utilizada e compreendida, uma vez que se trata de uma simplificação e encapsulamento de processos. Além disso, o PyMAPDL promove a integração do Ansys Mechanical com o python, o que torna o desenvolvimento de aplicações de Análise de Elementos Finitos (FEA) mais flexível.

6.1 Trabalhos futuros

Analisando os pontos não abordados neste trabalho, sugere-se:

- a investigação de outros métodos exatos da biblioteca SciPy;
- a investigação dos mesmos métodos exatos, mas implementados por outras bibliotecas;
- a investigação mais aprofundada de métodos aproximados, estocásticos e de otimização global, como o próprio AG;
- a investigação da otimização de novas configurações da junta, como, por exemplo, variar o chanfro interno e externo ou minimizar a quantidade de material (otimização topológica) enquanto o formato do chanfro varia;
- a investigação de métodos paralelos, a fim de acelerar o processo de otimização.

Referências

ALMEIDA, T. A. P. de. *Análise do efeito de filetes de adesivo na resistência estática de juntas adesivas tubulares*. 2020. 130 p. Dissertação (Mestrado em Engenharia Mecânica) — Instituto Superior de Engenharia do Porto, Porto, Portugal, 2020. Disponível em: <http://hdl.handle.net/10400.22/18190>. Acesso em: 06 mai. 2023. Citado na página 32.

ARAUJO, A. V. M. *Otimização Multiobjetivo e Simulação Computacional de Problema de Mecânica Estrutural*. 2022. 52 p. Dissertação (Bacharel em Engenharia de Computação) — Centro Federal de Educação Tecnológica de Minas Gerais, Timóteo, 2022. Disponível em: <https://drive.google.com/drive/folders/1gjcaQ5JJ7kGPJbDFlVQe7TfB8dTZBwxN>. Acesso em: 14 nov. 2023. Citado nas páginas 23, 25, 32 e 62.

BEIRANVAND, V.; HARE, W.; LUCET, Y. Best practices for comparing optimization algorithms. *Optimization and Engineering*, Springer Science and Business Media LLC, v. 18, n. 4, p. 815–848, sep 2017. Disponível em: <https://doi.org/10.1007%2Fs11081-017-9366-1>. Citado nas páginas 29 e 30.

BERASTEGUI, C. B. *Otimização de pórticos de aço e amortecedores viscosos sob excitação sísmica*. 2020. 87 p. Dissertação (Dissertação (Mestrado em Engenharia Mecânica)) — Universidade Federal do Rio Grande do Sul, Porto Alegre, Rio Grande do Sul, 2020. Disponível em: <http://hdl.handle.net/10183/214181>. Acesso em: 06 mai. 2023. Citado nas páginas 13 e 14.

BORGES, A. d. *Otimização de Forma e Paramétrica de Estruturas Treliçadas através dos Métodos Meta- Heurísticos Harmony Search e Firefly Algorithm*. 2013. 110 p. Dissertação (Mestrado em Programa de Pós-Graduação em Engenharia Mecânica) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2013. Citado na página 11.

BYRD, R. H. et al. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, v. 16, n. 5, p. 1190–1208, 1995. Disponível em: <https://doi.org/10.1137/0916069>. Citado na página 20.

CHAPRA, S.; CANALE, R. *Métodos Numéricos para Engenharia - 5.ed.* McGraw Hill Brasil, 2009. ISBN 9788580550115. Disponível em: <https://books.google.com.br/books?id=ch7MLP20q3MC>. Citado nas páginas 11 e 18.

CHAPRA, S. C.; CANALE, R. *Numerical Methods for Engineers*. 5. ed. USA: McGraw-Hill, Inc., 2005. ISBN 0073101567. Citado na página 30.

CHRISTENSEN, P.; KLARBRING, A. *An Introduction to Structural Optimization*. Springer Netherlands, 2008. (Solid Mechanics and Its Applications). ISBN 9781402086663. Disponível em: <https://books.google.com.br/books?id=80IeN__MYI8C>. Citado na página 13.

COELLO, C. Coello, a.c.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. comput. methods appl. mech. engrg. 191(11-12), 1245-1287. *Computer Methods in Applied Mechanics and Engineering*, v. 191, p. 1245–1287, 01 2002. Citado na página 23.

FERNANDES, F. J. S. *Otimização topológica do adesivo de juntas coladas constituídas pela sobreposição de dois aderentes utilizando o modelo de zona coesiva como critério de ruptura*. 2021. 236 p. Dissertação (Doutorado em Engenharia Mecânica)

— Universidade Estadual de Campinas, Campinas, Brasil, 2021. Disponível em: <https://repositorio.unicamp.br/acervo/detalhe/1232180>. Acesso em: 06 mai. 2023. Citado nas páginas 24 e 33.

FIRMINO, A. da S. *Métodos de otimização aplicados ao problema de recuperação de contêineres*. 2019. 115 p. Dissertação (Qualificação de Doutorado em Ciência da Computação) — Universidade Federal de Pernambuco, Recife, 2019. Disponível em: <https://repositorio.ufpe.br/handle/123456789/33479>. Acesso em: 06 mai. 2023. Citado na página 31.

GIL, J. *ANSYS Gets Into Open Source With GitHub*. 2021. Retrieved October 8, 201, from <https://www.ansys.com/blog/ansys-gets-into-open-source-with-github>. Citado na página 12.

GIVENS, J. A. H. G. H. Em optimization methods. In: ____. *Computational Statistics*. [s.n.], 2012. cap. 4, p. 97–126. ISBN 9781118555552. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118555552.ch4>. Citado na página 16.

GOLDBARG, M.; LUNA, H. *Otimização combinatória e programação linear: modelos e algoritmos*. Campus, 2000. ISBN 9788535205411. Disponível em: <https://books.google.com.br/books?id=QJuqtAEACAAJ>. Citado na página 14.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York: Addison-Wesley, 1989. Citado na página 22.

GOVEIA, T. de S. *Avaliação do desempenho mecânico de um epóxi reforçado com óxido de grafeno, sua comparação com adesivos comerciais de alta performance e aplicação simulada em juntas*. 2022. 75 p. Qualificação de Mestrado em Engenharia de Materiais — Centro Federal de Educação Tecnológica de Minas Gerais, Timóteo, Brasil, 2022. Citado nas páginas 11, 25, 26 e 27.

GOVEIA, T. de S. *SÍNTESE DE ÓXIDO DE GRAFENO PARA REFORÇO DE ADESIVO EPÓXI E MODELAGEM COMPUTACIONAL DA JUNTA ESTRUTURAL APRIMORADA*. 2023. 151 p. Dissertação de Mestrado — Centro Federal de Educação Tecnológica de Minas Gerais, Timóteo, Brasil, 2023. Citado nas páginas 25, 26 e 32.

GUIO, L. Z. *Modelo matemático e busca local aplicados ao problema de otimização de operação de pontes rolantes em um pátio de bobinas de aço*. 2021. 69 p. Dissertação (Dissertação (Mestrado em Computação Aplicada)) — Instituto Federal do Espírito Santo, Campus Serra, Serra, 2021. Disponível em: <https://repositorio.ifes.edu.br/handle/123456789/1285>. Acesso em: 06 mai. 2023. Citado na página 31.

HAFTKA, R.; GURDAL, Z. *Elements of Structural Optimization*. Springer Netherlands, 1991. (Solid Mechanics and Its Applications). ISBN 9780792315049. Disponível em: <https://books.google.com.br/books?id=mpwHaSWZklUC>. Citado nas páginas 13 e 14.

HILLIER, F. S.; LIEBERMAN, G. J. *Introduction to Operations Research*. 8. ed. New York, NY, USA: McGraw-Hill, 2010. Citado nas páginas 11 e 13.

INEGI. *Juntas adesivas híbridas são nova aposta para criar uniões mais resistentes*. 2020. Urlhttps://www.inegi.pt/pt/noticias/juntas-adesivas-hibridas-sao-nova- aposta-para-criar-unioes-mais-resistentes/. Citado na página 25.

KOZIEL, S.; YANG, X. (Ed.). *Computational Optimization, Methods and Algorithms*. Springer, 2011. v. 356. (Studies in Computational Intelligence, v. 356). ISBN 978-3-642-20858-4. Disponível em: <https://doi.org/10.1007/978-3-642-20859-1>. Citado na página 14.

LAGARIAS, J. C. et al. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on Optimization*, v. 9, n. 1, p. 112–147, 1998. Disponível em: <https://doi.org/10.1137/S1052623496303470>. Citado na página 15.

LEIGUS, A.; FENERICH, A. T.; MORAIS, M. de F. Aplicações da pesquisa operacional. *III Encontro de Engenharia de Produção Agroindustrial*, nov. 2009. Disponível em: <http://www.fecilcam.br/anais/iii_eepa/pdf/3_02.pdf>. Citado na página 13.

MELLO, M. F.; PEREIRA, R. O. A melhoria em processo produtivo com a utilização de um dispositivo semiautomatizado de dosagem e com eliminação de perdas. *XXXVI ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, 2016, JOÃO PESSOA. ENEGEP*, 2016. Disponível em: <https://abepro.org.br/biblioteca/TN_STO_226_323_28702.pdf>. Acesso em: 20 jun. 2023. Citado na página 11.

MENDES, D. A. *Optimizaçãoo*. 2010. Dissertação (Mestrado) — Instituto Universitário de Lisboa, 2010. Disponível em: <https://home.iscte-iul.pt/~deam/html/SlidesA3optim_2010.pdf>. Citado na página 17.

MORALES, J. L.; NOCEDAL, J. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, Association for Computing Machinery (ACM), v. 23, n. 4, p. 550–560, dez. 1997. ISSN 0098-3500. Citado na página 21.

MUNDSTOCK, D. C. *Otimização de forma utilizando o método dos elementos de contorno e cálculo de sensibilidade por variáveis complexas*. 2006. 91 p. Dissertação — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2006. Citado nas páginas 11 e 23.

NELDER, J. A.; MEAD, R. A Simplex Method for Function Minimization. *The Computer Journal*, v. 7, n. 4, p. 308–313, 01 1965. ISSN 0010-4620. Disponível em: <https://doi.org/10.1093/comjnl/7.4.308>. Citado na página 15.

NETO, M. A. da S. *gRPC: a tecnologia de chamada de procedimento remoto do Google para arquiteturas distribuídas*. 2022. Dissertação (Trabalho de Conclusão de Curso (Graduação em Tecnologia em Sistemas de Computação) — Instituto de Computação, Universidade Federal Fluminense, Niterói, 2022. Disponível em: <http://app.uff.br/riuff/handle/1/31580>. Citado na página 28.

NOCEDAL, J.; WRIGHT, S. J. *Numerical Optimization*. 2e. ed. New York, NY, USA: Springer, 2006. Citado na página 14.

NUNES, A. F. da S. *Otimização da escala de trabalho de policiais penais no Rio Grande do Su*. 2022. 84 p. Dissertação (Dissertação (Mestrado em Administração)) — Universidade Federal do Rio Grande do Su, Porto Alegre, Brasil, 2022. Disponível em: <http://hdl.handle.net/10183/246520>. Acesso em: 06 mai. 2023. Citado na página 32.

OCANA, R. et al. Evaluation of degradation of structural adhesive joints in functional automotive applications. *Procedia Engineering*, v. 132, p. 716–723, 2015. ISSN 1877-7058. MESIC Manufacturing Engineering Society International Conference 2015. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877705815044641>. Citado na página 25.

OLIVEIRA, M. M. de. *Estudo sobre a otimização de parâmetros de fabrico de juntas adesivas de simples sobreposição para maximizar a resistência mecânica*. 2021. 63 p. Dissertação (Mestrado em Engenharia Industrial) — Instituto Politécnico de Bragança, Bragança, Portugal, 2021. Disponível em: <https://bibliotecadigital.ipb.pt/handle/10198/25751>. Acesso em: 06 mai. 2023. Citado na página 33.

PEDERSEN, P. Optimal joint positions for space trusses. *Journal of the Structural Division*, v. 99, n. 12, p. 2459–2476, 1973. Disponível em: <https://ascelibrary.org/doi/abs/10.1061/JSDEAG.0003669>. Citado na página 24.

PLAGIANAKOS, V.; VRAHATIS, M. Advances in combinatorial and global optimization. In: MIGDALAS, A.; PARDALOS, P.; BURKARD, R. (Ed.). *Advances in Combinatorial and Global Optimization*. River Edge: World Scientific, 2001. p. 283–296. Citado na página 52.

POWELL, M. J. D. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, v. 7, n. 2, p. 155–162, 01 1964. ISSN 0010-4620. Disponível em: <https://doi.org/10.1093/comjnl/7.2.155>. Citado na página 17.

POWELL, M. J. D. A direct search optimization method that models the objective and constraint functions by linear interpolation. In: ____. *Advances in Optimization and Numerical Analysis*. Dordrecht: Springer Netherlands, 1994. p. 51–67. ISBN 978-94-015-8330-5. Disponível em: <https://doi.org/10.1007/978-94-015-8330-5_4>. Citado nas páginas 18 e 19.

PYANSYS. *DOCUMENTAÇÃO PyMAPDL 0.67.0.* 2023. Accessed: 2010-09-30. Disponível em: <https://mapdl.docs.pyansys.com/version/stable/#documentation-and-issues>. Acesso em: 19 nov. 2023. Citado nas páginas 28 e 29.

REIS, M. O. dos. *Desenvolvimento e caracterização de um adesivo epóxi nanomodificado e avaliação de desempenho das juntas adesivadas*. 2022. 220 p. Tese (Doutorado em Engenharia Mecânica) — Universidade Federal de Minas Gerais, Belo Horizonte, 2022. Disponível em: <http://hdl.handle.net/1843/49388>. Citado na página 25.

ROTHLAUF, F. Design principles. In: ____. *Design of Modern Heuristics: Principles and Application*. Berlin, Heidelberg: "Springer Berlin Heidelberg, 2011. p. 157–171. ISBN 978-3-540-72962-4. Disponível em: <https://doi.org/10.1007/978-3-540-72962-4_6>. Citado na página 11.

SCHON, C. G. *Mecânica dos materiais: fundamentos e tecnologia do comportamento mecânico*. [S.l.]: Elsevier, 2013. Citado nas páginas 27 e 43.

SCHäLTE, Y.; STAPOR, P.; HASENAUER, J. Evaluation of derivative-free optimizers for parameter estimation in systems biology. In: *IFAC PapersOnLine*. [S.l.]: Elsevier B.V., 2018. v. 51, p. 98–101. ISSN 24058963. Citado na página 31.

SORITZ, S.; MOSER, D.; GRUBER-WöLFLER, H. Comparison of derivative-free algorithms for their applicability in self-optimization of chemical processes. *Chemistry-Methods*, John Wiley and Sons Inc, v. 2, 5 2022. ISSN 26289725. Citado na página 31.

SOUZA, L. A. P. *Comparação entre métodos de otimização aplicados aos ajustes de relés direcionais de sobrecorrente*. 2019. 135 p. Dissertação (Mestrado em Engenharia Elétrica e da Computação) — Universidade Federal de Goiás, Goiânia, 2019. Disponível em: <http://repositorio.bc.ufg.br/tede/handle/tede/9667>. Acesso em: 14 nov. 2023. Citado na página 31.

STUMPF, F. T.; GöTZ, G. L.; LEON, D. M. D. Strain- and stress-based parametric optimization of fiber-reinforced elastomers under finite deformations. *Mechanics of Materials*, Elsevier B.V., v. 179, 4 2023. ISSN 01676636. Citado na página 32.

SURJANOVIC, S.; BINGHAM, D. *Virtual Library of Simulation Experiments: Test Functions and Datasets*. 2015. Retrieved December 4, 2023, from <http://www.sfu.ca/~ssurjano>. Citado nas páginas 38 e 39.

VAROQUAUX, G. *Mathematical optimization: finding minima of functions*. 2017. Disponível em: <https://scipy-lectures.org/advanced/mathematical_optimization/index.html>. Citado na página 35.

VELEZ, D. I. A. *Otimização de estruturas reticuladas utilizando algoritmos genéticos*. 2015. 115 p. Dissertação (Mestrado em Estruturas e Construção Civil — Universidade de Brasília, Brasília, 2015. Disponível em: <https://repositorio.unb.br/handle/10482/19863>. Citado na página 25.

WRIGHT, M. Direct search methods: Once scorned, now respectable. In: ____. *Numerical analysis*. [S.l.]: Addison-Wesley, 1996. p. 191–208. Citado na página 15.

Apêndices

APÊNDICE A – Tabelas dos testes com as funções de estudo

Nesta seção estão as tabelas com os resultados das execuções de cada algoritmo, divididas conforme a função objetivo utilizada.

Tabela 23 – Resultados dos métodos exatos

Função Rosenbrock | Valor mínimo $f(x) = 0$

Método	**F(x)**	**Tempo (s)**	**Desvio Padrão**	**Erro Absoluto**
Nelder-Mead	0	0,55	0,01	0
Powell	0	0,04	0	0
COBYLA	0	0,17	0	0
L-BFGS-B	0	0,02	0,06	0

Função Sphere | Valor mínimo $f(x) = 0$

Método	**F(x)**	**Tempo (s)**	**Desvio Padrão**	**Erro Absoluto**
Nelder-Mead	0,02	2,66	0,35	0,02
Powell	0	0,02	0	0
COBYLA	0	0,14	0,01	0
L-BFGS-B	0	0,01	0	0

Função Ackley | Valor mínimo $f(x) = 0$

Método	**F(x)**	**Tempo (s)**	**Desvio Padrão**	**Erro Absoluto**
Nelder-Mead	3,57	0,53	0,02	3,57
Powell	3,57	0,10	0,01	3,57
COBYLA	3,57	0,18	0,02	3,57
L-BFGS-B	3,57	0,03	0,04	3,57

Fonte: o autor

Na Tabela 23:

- a coluna “F(x)” corresponde ao valor final da função após a otimização;
- a coluna “Tempo” corresponde a média dos tempos (em segundos) das 10 execuções de cada método;
- a coluna “Desvio Padrão” corresponde ao desvio padrão das 10 execuções de cada método.

- a coluna “Erro Absoluto” corresponde ao erro absoluto do valor de “F(x)”. É calculado pela fórmula 2.6.

Tabela 24 – Resultados do método aproximado

Função Rosenbrock | Valor mínimo $f(x) = 0$

Método	**F(x)**	**Tempo (s)**	**Desvio Padrão**	**Erro Absoluto**
AG	0	1,66	0,13	0

Função Sphere | Valor mínimo $f(x) = 0$

Método	**F(x)**	**Tempo (s)**	**Desvio Padrão**	**Erro Absoluto**
AG	0	0,90	0,09	0

Função Ackley | Valor mínimo $f(x) = 0$

Método	**F(x)**	**Tempo (s)**	**Desvio Padrão**	**Erro Absoluto**
AG	0,44	1,60	0,16	0,44

Fonte: o autor

Na Tabela 24:

- a coluna “F(x)” corresponde ao menor valor final da função obtido após 10 execuções;
- a coluna “Tempo” corresponde a média dos tempos (em segundos) das 10 execuções de cada método;
- a coluna “Desvio Padrão” corresponde ao desvio padrão das 10 execuções de cada método;
- a coluna “Erro Absoluto” corresponde ao erro absoluto do valor de “F(x)”. É calculado pela fórmula 2.6.

APÊNDICE B – Tabelas dos testes de variação de parâmetros

Nesta seção estão as tabelas com os resultados das execuções de cada método com variação de parâmetro.

Tabela 25 – Resultados do método Powell

direc = [0,1],[1,0]

Rodada	**F(x) (MPa)**	**Tempo (s)**
1	599,48	785,81
2	615,37	760,89
3	620,35	731,65
4	620,73	636,87
5	609,14	728,60
Média	613,01	728,60
Desvio Padrão	8,91	50,48

direc = [1,0],[0,1]

Rodada	**F(x) (MPa)**	**Tempo (s)**
1	613,79	652,61
2	609,43	654,79
3	613,14	838,36
4	610,71	640,86
5	609,10	650,65
Média	611,23	687,45
Desvio Padrão	2,14	84,53

direc = [0,2],[2,0]

Rodada	**F(x) (MPa)**	**Tempo (s)**
1	612,24	655,26
2	617,67	688,80
3	612,75	685,67
4	609,35	703,18
5	609,55	741,63
Média	612,31	694,91
Desvio Padrão	3,36	31,41

direc = none

Rodada	F(x) (MPa)	Tempo (s)
1	603,78	965,43
2	608,08	884,70
3	591,70	619,07
4	604,55	752,87
5	598,20	953,75
Média	602,46	829,76
Desvio Padrão	4,13	151,64

Fonte: o autor

Tabela 26 – Resultados do método Nelder-Mead

initial_simplex = [1,0], [0,1], [-1,0]

Rodada	F(x) (MPa)	Tempo (s)
1	678,64	133,38
2	678,64	146,26
3	678,64	134,09
4	678,64	125,48
5	678,64	159,50
Média	678,64	139,74
Desvio Padrão	0	13,318

initial_simplex = [2,2], [7,7], [10,10]

Rodada	F(x) (MPa)	Tempo (s)
1	678,64	77,07
2	678,64	80,44
3	678,64	69,10
4	678,64	70,01
5	678,64	79,24
Média	678,64	75,17
Desvio Padrão	0	5,28

initial_simplex = [20,10], [15,20], [10,30]

Rodada	F(x) (MPa)	Tempo (s)
1	592,08	181,74
2	592,08	168,49
3	592,08	184,56
4	592,08	185,36
5	592,08	169,96

Média	592,08	178,02
Desvio Padrão	0	8,16

initial_simplex = none

Rodada	**F(x) (MPa)**	**Tempo (s)**
1	592,08	286,13
2	592,08	341,84
3	592,08	284,55
4	592,08	287,84
5	592,08	332,41
Média	592,08	299,35
Desvio Padrão	0	38,04

Fonte: o autor

Tabela 27 – Resultados do método COBYLA

tol = 1e-2 e catol = 1e-2

Rodada	**F(x) (MPa)**	**Tempo (s)**
1	605,54	568,21
2	676,47	207,48
3	685,94	203,62
4	682,879	182,75
5	672,55	164,62
Média	664,68	265,33
Desvio Padrão	33,47	170,19

tol = 1e-5 e catol = 1e-5

Rodada	**F(x) (MPa)**	**Tempo (s)**
1	646,35	551,10
2	663,72	528,28
3	675,14	491,05
4	681,84	524,92
5	672,19	465,49
Média	667,85	512,17
Desvio Padrão	13,67	33,78

tol = 1e-10 e catol = 1e-10

Rodada	**F(x) (MPa)**	**Tempo (s)**
1	669,93	753,24
2	676,60	817,82

3	673,99	829,84
4	670,00	738,75
5	681,60	736,15
Média	674,42	775,16
Desvio Padrão	4,91	45,11

tol = none e catol= none

Rodada	**F(x) (MPa)**	**Tempo (s)**
1	680,04	356,47
2	676,75	349,73
3	648,84	345,12
4	676,46	374,90
5	681,09	339,63
Média	672,64	353,19
Desvio Padrão	13,45	13,62

Fonte: o autor

Tabela 28 – Resultados do método L-BFGS-B

maxls = 1000

Execução	**F(x) (MPa)**	**Tempo (s)**
1	686,62	759,46
2	684,31	714,04
3	689,65	983,62
4	690,85	953,72
5	687,46	981,73
Média	687,78	878,52
Desvio Padrão	2,57	130,94

Fonte: o autor

Tabela 29 – Resultados do método AG

pop_size = 50 e n_gen = 10 e eliminate_duplicates = True

Execução	**F(x) (MPa)**	**Tempo (s)**
1	621,44	135,24
2	692,61	172,02

3	679,59	172,65
4	666,72	142,67
5	650,55	134,42
Média	662,18	151,40
Desvio Padrão	27,59	19,38

Fonte: o autor

APÊNDICE C – Código-fonte da otimização do problema *benchmark*

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
from scipy import optimize
from ansys.mapdl.core import launch_mapdl
mapdl = launch_mapdl(loglevel="WARNING", print_com=True)


# In[2]:


_FIG_PATH = "figuras/"

MAT_ADV = 1 # Identificador do material do adesivo
MAT_ADH = 2 # Identificador do material do aderente

# Propriedades da estrutura
T_ADV=0.2 # Espessura do adesivo (mm)
T_ADH=1.6 # Espessura do aderente (mm)
L_ADH=100    # Comprimento do aderente (mm)
WIDTH=25     # Largura do aderente (mm)
RESPIRO = 0.05 # Margem adicionada aos keypoints para evitar a sobreposi  o

# Propriedades dos materiais
E_ADV=2522 # M dulo de Young do adesivo (N/mm2)
NU_ADV=0.4  # Coeficiente de Poisson do adesivo
E_ADH=70e3  # M dulo de Young do aderente (N/mm2)
NU_ADH=0.33  # Coeficiente de Poisson do aderente

LOAD=5000  # Define total applied load in N
_QTD_CURVA = 20 #Quantidade de curvas de n vel no gr fico
LIMITE_CHANFRO = 15 # Quantidade chanfros analisados


# In[3]:


def pre_processamento(chanfro, sobreposicao):
    mapdl.prep7()  # Entra no modo de pr -processamento
```

```
46      mapdl.et(1, "82", "", "", 2) # Define o elemento 82 (com estado plano de
            deforma  o)
47
48      # Define as propriedades dos materiais
49      mapdl.mp("ex", MAT_ADV, E_ADV)
50      mapdl.mp("nuxy", MAT_ADV, NU_ADV)
51      mapdl.mp("ex", MAT_ADH, E_ADH)
52      mapdl.mp("nuxy", MAT_ADH, NU_ADH)
53
54      # Defini  o da geometria
55      # Keypoints
56      mapdl.k(1, 0, 0, 0)
57      mapdl.k(2, L_ADH-chanfro, 0, 0)
58      mapdl.k(3, 0, T_ADH, 0)
59      mapdl.k(4, L_ADH, T_ADH, 0)
60      mapdl.k(5, L_ADH-sobreposicao, T_ADH, 0)
61      mapdl.k(6, L_ADH, T_ADH+T_ADV, 0)
62      mapdl.k(7, L_ADH-sobreposicao, T_ADH+T_ADV, 0)
63      mapdl.k(8, 2*L_ADH-sobreposicao, T_ADH+T_ADV, 0)
64      mapdl.k(9, L_ADH-sobreposicao+chanfro, 2*T_ADH+T_ADV, 0)
65      mapdl.k(10, 2*L_ADH-sobreposicao, 2*T_ADH+T_ADV, 0)
66
67      #  reas
68      mapdl.a(1, 2, 4, 5, 3)  # Aderente inferior
69      mapdl.a(5, 4, 6, 7)  # Adesivo
70      mapdl.a(7, 6, 8, 10, 9)  # Aderente superior
71
72      mapdl.esize(T_ADV)  # Define o tamanho m nimo geral da malha
73      mapdl.aesize(2, T_ADV/3) # Define o tamanho m nimo da malha para o adesivo
74      mapdl.lesize(2, T_ADV/3) # Linha do chanfro inferior
75      mapdl.lesize(12, T_ADV/3) # Linha do chanfro superior
76
77      mapdl.mat(MAT_ADV)  # Aplica as propriedades do material do adesivo
78      mapdl.mat(MAT_ADH)  # Aplica as propriedades do material do aderente
79
80      mapdl.amesh(2)  # Gera a malha da  res  do adesivo
81      mapdl.amesh(1, 3, 2)  # Gera a malha da  res  dos aderentes
82
83      mapdl.krefine(5, 6, 1, 2, 1) # Refina a malha em torno das singularidades (
            keypoints 5 e 6)
84      mapdl.finish()  # Sai do modo de pr -processamento
85
86  def processamento(sobreposicao):
87      mapdl.slashsolu()  # Entra no modo de solu  o
88
89      #Aplica  o das condi  es de contorno
90      #Engaste no aderente inferior
91      mapdl.ksel("s", "loc", "x", 0) # Seleciona os keypoints em x=0
92      mapdl.ksel("r", "loc", "y", 0) # Mant m a sele  o e seleciona tamb m os
            keypoints em y=0
93      mapdl.dk("all", "all")  # Restringe o deslocamento de todos os keypoints
            selecionados (no caso 1) em todas as dire  es
```

```
94
95      #Apoio no aderente superior
96      mapdl.ksel("s", "loc", "x", 2*L_ADH-sobreposicao) # Seleciona os keypoints em x
            =2lt-sobreposicao, Aresta    direita do substrato superior
97      mapdl.ksel("r", "loc", "y", T_ADH+T_ADV) # Mant m a sele  o e seleciona os
            keypoints em y= T_ADH+T_ADV
98      mapdl.dk("all", "uy") # Restringe o deslocamento de todos os keypoints
            selecionados (no caso 1) na dire  o Y
99
100     #Press o nas faces das extremidades externas dos aderentes
101     mapdl.ksel("s", "loc", "x", 2*L_ADH-sobreposicao) # Seleciona os keypoints em x
            =2lt-sobreposicao, Aresta    direita do substrato superior
102     mapdl.ksel("a", "loc", "x", 0) # Adiciona    sele  o os keypoints em x=0,
            Aresta    esquerda do substrato inferior
103     mapdl.lslk("s", 1)  # Seleciona as linhas que cont m os keypoints selecionados
104     mapdl.sfl("all", "pres", -LOAD/(T_ADH*WIDTH)) # Aplica a press o na linha
            selecionada (p=-LOAD/(T_ADH*WIDTH))
105
106     mapdl.ksel("all")  # Seleciona todos os keypoints
107     mapdl.lsel("all")  # Seleciona todas as linhas
108     mapdl.solve()  # Resolve o modelo
109     mapdl.finish() # Sai do modo de processamento
110
111 def pos_processamento():
112     mapdl.post1()  # Entra no modo de P s processamento
113
114     # Imprime os resultados calculados relevantes
115     mapdl.eplot(cpos="xy") # Plota os elementos
116     mapdl.post_processing.plot_nodal_component_stress('Y', cpos="xy", show_edges=
            False) # Plota a distribui  o tens es nodais normais
117
118     mapdl.result.plot_principal_nodal_stress(0,'seqv',background='w',show_edges=True
            ,text_color='k',add_text=True, cpos="xy")
119     mapdl.post_processing.plot_nodal_eqv_stress(cpos="xy", show_edges=True)
120     mapdl.finish() # Sai do modo de pos processamento
121
122 def fem(params):
123     chanfro = params[0]
124     sobreposicao = params[1]
125     mapdl.clear()
126     pre_processamento(chanfro, sobreposicao)
127     processamento(sobreposicao)
128
129
130 def nodal_stress_stiffness_displacement(params):
131     fem(params)
132     chanfro, sobreposicao = params
133     max_disp_y = np.max(mapdl.post_processing.nodal_displacement('Y'))
134     min_stiffness = LOAD/np.max(mapdl.post_processing.nodal_displacement('X'))
135     max_stress = np.max(mapdl.post_processing.nodal_component_stress('Y'))
136     print("Chanfro:", chanfro, "\tSobreposicao:", sobreposicao, "\tTens oY:",
            max_stress, "\tRigidez:", min_stiffness)
```

```
137     #print(params, max_stress, min_stiffness, max_disp_y)
138     return max_stress, min_stiffness, max_disp_y
139
140
141 def plot_deformada():
142     mapdl.post1()
143     mapdl.allsel()
144     mapdl.set('last')
145     mapdl.upcoord(1)
146     mapdl.eplot(cpos='xy')
147     mapdl.upcoord(-1)
148     mapdl.finish()
149
150
151 # In[13]:
152
153
154 chanfros = [0, 2, 4, 6, 8, 10, 12, 14, 15]
155 sobreposicoes = [45, 40, 35, 30, 25]
156 stress = []
157 displacement_y = []
158 stiffness = []
159 for i_sobp in range(len(sobreposicoes)):
160     stress.append([])
161     displacement_y.append([])
162     stiffness.append([])
163     for chanfro in chanfros:
164         max_stress, min_stiffness, max_disp_y = nodal_stress_stiffness_displacement
                ([chanfro, sobreposicoes[i_sobp]])
165         stress[i_sobp].append(max_stress)
166         displacement_y[i_sobp].append(max_disp_y)
167         stiffness[i_sobp].append(min_stiffness)
168
169 str_date_time = datetime.now().strftime("%Y%m%d%H%M%S")
170 x,y = np.meshgrid(chanfros, sobreposicoes)
171
172 plt.figure("M xima Tens o em Y")
173 plt.contour(x,y,stress,_QTD_CURVA)
174 plt.colorbar(label='Tens o (MPa)')
175 plt.ylabel('Sobreposi  o (mm)')
176 plt.xlabel('Chanfro (mm)')
177 plt.savefig(_FIG_PATH+"tensao_y_2D_"+str_date_time+".png")
178 plt.show()
179
180 plt.figure("M ximo Deslocamento em Y")
181 plt.contour(x,y,displacement_y,_QTD_CURVA)
182 plt.colorbar(label='Deslocamento em Y (mm)')
183 plt.ylabel('Sobreposi  o (mm)')
184 plt.xlabel('Chanfro (mm)')
185 plt.savefig(_FIG_PATH+"deslocamento_y_2D_"+str_date_time+".png")
186 plt.show()
187
```

```
plt.figure("M nima rigidez")
plt.contour(x,y,stiffness,_QTD_CURVA)
plt.colorbar(label='Rigidez (N/mm)')
plt.ylabel('Sobreposi o (mm)')
plt.xlabel('Chanfro (mm)')
plt.savefig(_FIG_PATH+"rigidez_2D_"+str_date_time+".png")
plt.show()


# In[7]:


import timeit

chute=[10, 30]
bounds=[(0, LIMITE_CHANFRO), (25, 45)]

def tensao(params):
    chanfro, sobreposicao = params
    max_stress, _, _ = nodal_stress_stiffness_displacement([chanfro, sobreposicao])
    return max_stress


# ## Powell

# In[7]:


tempo_inicio = timeit.default_timer()
sol_powell = optimize.minimize(tensao, chute, method="Powell", bounds=bounds,
    options={'xtol': 1e-1, 'maxfev': 1e+20})
tempo_fim = timeit.default_timer()

tempo = tempo_fim - tempo_inicio
print("Tempo total (s): ", tempo)

sol_powell


# ## COBYLA

# In[8]:


# chanfro >= 0
def restricao1(params):
    chanfro, sobreposicao = params
    return chanfro

# chanfro <= 15
def restricao2(params):
    chanfro, sobreposicao = params
```

```
239     return 15 - chanfro
240
241 # sobreposicao >= 25
242 def restricao3(params):
243     chanfro, sobreposicao = params
244     return sobreposicao - 25
245
246 # sobreposicao <= 45
247 def restricao4(params):
248     chanfro, sobreposicao = params
249     return 45 - sobreposicao
250
251 constraints = [
252     {'type': 'ineq', 'fun': restricao1},
253     {'type': 'ineq', 'fun': restricao2},
254     {'type': 'ineq', 'fun': restricao3},
255     {'type': 'ineq', 'fun': restricao4},
256 ]
257
258 tempo_inicio = timeit.default_timer()
259 sol_cobyla = optimize.minimize(tensao, chute, method="COBYLA", constraints=
        constraints, options={'catol':1e-10, 'tol':1e-10})
260 tempo_fim = timeit.default_timer()
261
262 tempo = tempo_fim - tempo_inicio
263 print("Tempo total (s): ", tempo)
264
265 sol_cobyla
266
267
268 # ## Nelder-Mead
269
270 # In[9]:
271
272
273 tempo_inicio = timeit.default_timer()
274 sol_nealder = optimize.minimize(tensao, chute, method="Nelder-Mead", bounds=bounds,
        options={'xatol': 1e-1, 'maxfev': 1e+5})
275 tempo_fim = timeit.default_timer()
276
277 tempo = tempo_fim - tempo_inicio
278 print("Tempo total (s): ", tempo)
279
280 sol_nealder
281
282
283 # ## L-BFGS-B
284
285 # In[10]:
286
287
288 tempo_inicio = timeit.default_timer()
```

```
sol_bfgs = optimize.minimize(tensao, chute, bounds=bounds, method='L-BFGS-B',
    options={'maxls': 100000})
tempo_fim = timeit.default_timer()

tempo = tempo_fim - tempo_inicio
print("Tempo total (s): ", tempo)

sol_bfgs


# In[4]:


from pymoo.algorithms.moo.nsga2 import NSGA2
from pymoo.algorithms.soo.nonconvex.ga import GA
from pymoo.algorithms.soo.nonconvex.pso import PSO
from pymoo.optimize import minimize
from pymoo.visualization.scatter import Scatter
from pymoo.core.problem import Problem

from pymoo.operators.crossover.pntx import TwoPointCrossover
from pymoo.operators.mutation.bitflip import BitflipMutation
from pymoo.operators.sampling.rnd import BinaryRandomSampling


# In[27]:


pop = 100
n_gen = 200

class problem(Problem):
    def __init__(self):
        super().__init__(n_var=2, n_obj=1, xl=[0, 25], xu=[15,45])

    def _evaluate(self, x, out, *args, **kwargs):
        out = tensao([x[0][0], x[0][1]])


# ## NSGA-II

# In[18]:


algorithm = NSGA2(pop_size=pop, crossover=TwoPointCrossover(),  eliminate_duplicates
    =True)

tempo_inicio = timeit.default_timer()
res = minimize(problem(), algorithm, ('n_gen', n_gen))
tempo_fim = timeit.default_timer()

tempo = tempo_fim - tempo_inicio
```

```
print("Tempo total (s): ", tempo)


## AG

In[28]:


algorithm = GA(pop_size=pop, crossover=TwoPointCrossover(), eliminate_duplicates=
    True)

tempo_inicio = timeit.default_timer()
res = minimize(problem(), algorithm, ('n_gen', n_gen))
tempo_fim = timeit.default_timer()

tempo = tempo_fim - tempo_inicio
print("Tempo total (s): ", tempo)


In[31]:


nodal_stress_stiffness_displacement([res.X[0], res.X[1]])


In[30]:


res.X[0], res.X[1]


In[ ]:


mapdl.exit() #Encerra o mapdl
```

slj_functions_2d.py