XXXII Congresso de Iniciação Científica da UNICAMP – 2024

Otimização topológica de estruturas de grande escala sob cargas de peso próprio

Beatriz Bragaglia e Prof. Dr. Josué Labaki

FEM, UNICAMP

Resumo

O trabalho discute a aplicação da otimização topológica (OT) em estruturas, destacando sua importância no contexto atual de recursos limitados, preocupações ambientais e competição tecnológica. A OT busca encontrar layouts ideais de estruturas, minimizando uma função objetivo, como o estado de tensão ou compliance, sujeita a condições de projeto. São revisados alguns métodos de OT, como BESO, LSTO, SIMP e TOBS, ressaltando seu impacto econômico e ambiental positivo, especialmente em grandes estruturas. Um dos desafios enfrentados é lidar com forças de corpo, como o peso da estrutura, que podem levar a soluções triviais. Para superar esse desafio, propõe-se penalizar a rigidez de cada elemento, além de impor restrições de massa e volume. O foco é na resolução do problema de OT usando programação linear inteira (ILP) através do método TOBS. O trabalho aborda essencialmente a aplicação da OT em estruturas tridimensionais, incluindo cargas de peso-próprio. A metodologia inclui a formulação do problema de otimização, o uso da ILP, filtragem e análise de sensibilidade, considerando cargas de peso-próprio como forças de corpo. Por fim, discute-se alguns estudos de caso para entendimento prático do assunto.

Palavras-chave: *Otimização Topológica, Interação Solo-Estrutura, Peso-próprio*

1. Introdução

Observando o contexto de produção atual, vemos que cada vez mais recursos estão limitados, a preocupação com o baixo impacto ambiental aumenta e a competição tecnológica se intensifica. Nesse cenário, a otimização topológica, uma ferramenta de design de estruturas, tem ganhado cada vez mais popularidade. Essa ferramenta resume-se a encontrar layouts de estruturas ideais, minimizando uma função objetivo, como o estado de tensão ou a *compliance* da estrutura, sujeito a condições de projeto, como condições de volume. Com isso, o projeto estrutural pode ser adaptado de forma otimizada para atender às demandas do problema, que economiza despesas e o uso de materiais, melhora o comportamento deformacional, isto é, torna a estrutura mais rígida ao minimizar a *compliance* e reduz o tempo de construção. Além disso, o problema de otimização topológica não precisa de uma solução prévia, podendo ser possível gerar uma estrutura otimizada a partir de um simples bloco e as condições de contorno da estrutura em questão, como engastes, apoios e os carregamentos envolvidos.

Desde Bendsøe e Kikuchi (1998) e mais precisamente após Deaton e Grandi (2014), alguns métodos de otimização topológica surgiram, entre eles podemos citar o BESO, LSTO, SIMP e TOBS. O desenvolvimento desses métodos possibilitou a criação de softwares comerciais que permitem realizar esse tipo de análise, como o Ansys, COMSOL e Altair.

Os impactos econômicos e ambientais positivos gerados pela utilização da OT podem ser sentidos mais fortemente em estruturas de grande porte, como torres, prédios e pontes. Entretanto, uma dificuldade que é classicamente encontrada nesse tipo de problema é que a carga externa, que é principalmente o peso da estrutura, está uniformemente distribuído por todos os elementos da estrutura em termos de nós equivalentes. Em um problema em que o objetivo é maximizar a rigidez da estrutura, o otimizador é levado a remover o máximo de elementos possíveis, já que a remoção de elementos é igual ao alívio da carga, gerando uma estrutura mais rígida. Contudo, geraria uma solução trivial e estamos interessados em uma solução não nula. Por isso, propomos superar esta dificuldade penalizando a rigidez de cada elemento e as restrições de massa e volume com funções separadas (Huang e Xie, 2011).

Para resolver o problema de otimização, vamos utilizar variáveis de projeto binárias, definindo um conjunto de variáveis de dimensionamento 0,1, onde 0 significa a ausência de material e 1 significa a localização do material sólido. Todavia, o problema binário é considerado muito difícil de se resolver utilizando programação matemática formal. A primeira tentativa de resolver problemas binários de OT foi proposta por Xie e Steven (1997), partindo da ideia de que materiais ineficientes poderiam ser retirados da estrutura, o método ficou conhecido como ESO (Evolutional Structure Optimization). Logo depois, foram feitas tentativas para desenvolver uma versão bidirecional do ESO, em que material também poderia ser adicionado à estrutura. Hoje conhecemos o BESO após Huang e Xie (2007) apresentarem soluções convergentes e independentes de malha. O BESO utiliza uma restrição de volume para sistematizar o a atualização da estrutura, entretanto isso implica que uma restrição de volume deve estar sempre presente na formulação de otimização, impedindo que problemas com restrição de massa ou de múltiplas restrições sejam resolvidos. Portanto, faz sentido buscarmos incluir a programação matemática em problemas de OT.

No nosso caso, estaremos realizando otimização topológica de estruturas binárias utlizando programação linear inteira, a partir do método TOBS, desenvolvido por Sivapuram e Picelli (2020). O TOBS combina os recursos de análise de sensibilidade e filtragem inde-

pendente de malha do método BESO desenvolvido por Huang e Xie (2007) e a programação linear inteira sequencial para otimização de topologia discreta desenvolvida por Svanberg e Werme (2006). O TOBS resolve o problema apresentado no BESO, agora podemos colocar restrições que não são volumétricas e não precisamos utilizar Multiplicadores de Lagrange para isso, como propos Beckers (1999). Este método utiliza programação matemática a partir de um algoritmo Branch-and-bround, com isso, a classificação de números de sensibilidade e os limites não são usados para atualizar as variáveis de projeto.

No problema de estruturas de larga escala não incorporaremos as estacas das mesmas, entretanto já é conhecido os efeitos da interação com o solo na OT. Estaremos incorporando uma carga de peso-próprio distribuída ao longo do corpo, modelando uma força de corpo, em vez de uma carga concentrada em um único ponto. Para tal, ampliaremos o método TOBS para estruturas tridimensionais e incorporando a força de corpo relativa ao peso-próprio da estrutura. Depois faremos um estudo de caso para observar o que ocorre com a estrutura se alterarmos alguns parâmetros, como o sentido do carregamento e a redução da penalização da rigidez elementar.

2. Metodologia e Formulação

O problema de otimização consiste em minimizar uma função $f(x)$ sujeita à algumas condições, nesse caso nossa função a ser otimizada será a compliance estrutural. Onde K é a matriz de rigidez global, u é o campo de deslocamentos estruturais, $V(x)$ é o volume da estrutura, V_0 é o volume inicial, $\overline{V}$ é o volume final mínimo que a estrutura deve ter, que é definido pelo usuário, x_j são as entradas do vetor de variáveis de projeto, nesse caso trata-se das densidades, que possui um tamanho N_d. Vamos utilizar Programação Linear Inteira para resolver esse problema de minimização.

2.1. Programação Linear Inteira

Um problema geral de otimização é um problema com inifitas variáveis de projeto, uma forma para tornar esse problema passível de ser tratado computacionalmente é aproximar as variáveis de projeto x_j. Nesse caso, estamos assumindo que a escolha do material é uniforme em cada elemento finito no domínio do projeto. É a partir disso que obtemos a seguinte formulação.

$$\begin{aligned} \underset{x}{\text{Minimizar}} \quad & f(x) \\ \text{Sujeito à} \quad & g_i(\boldsymbol{x}) \leq \overline{g}_i,\ i \in [1, N_g] \\ & x_j \in \{0,1\},\ j \in [1, N_d] \end{aligned}$$

Onde $f(x)$ é a função objetivo, no nosso caso é a *compliance*, g_i são as restrições de desigualdade de tamanho N_g, no problema em 2.1 é uma restrição de volume.

Se utilizarmos uma aproximação pela série de Taylor, vamos obter as seguintes expressões (1) e (2) para as funções objetivo e restrição. Onde $O(\left\|\Delta \boldsymbol{x}^k\right\|_2^2)$ é o erro de truncamento.

$$f(\boldsymbol{x}) = f(\boldsymbol{x}^k) + \frac{\partial f(\boldsymbol{x}^k)}{\partial \boldsymbol{x}} \cdot \Delta \boldsymbol{x}^k + O(\left\|\Delta \boldsymbol{x}^k\right\|_2^2), \tag{1}$$

$$g_i(\boldsymbol{x}) = g_i(\boldsymbol{x}^k) + \frac{\partial g_i(\boldsymbol{x}^k)}{\partial \boldsymbol{x}} \cdot \Delta \boldsymbol{x}^k + O(\left\|\Delta \boldsymbol{x}^k\right\|_2^2), \tag{2}$$

Utilizamos essa aproximação porque os problemas de OT são não lineares e não convexos, o método que estamos utilizando gera problemas de subotimização linear inteira aproximada sequencialmente e resolve os programas lineares inteiros gerados.

Logo, podemos aproximar da seguinte maneira

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}^k) + \frac{\partial f(\boldsymbol{x}^k)}{\partial \boldsymbol{x}} \cdot \Delta \boldsymbol{x}^k \tag{3}$$

$$g_i(\boldsymbol{x}) \approx g_i(\boldsymbol{x}^k) + \frac{\partial g_i(\boldsymbol{x}^k)}{\partial \boldsymbol{x}} \cdot \Delta \boldsymbol{x}^k \tag{4}$$

Obtendo na forma unificada:

$$\Delta {x_j}^k \in \{-{x_j}^k, 1 - {x_j}^k\} \tag{5}$$

E então o otimizador escolhe um Δx_j^k ideal para manter o erro de truncamento $O(\left\|\Delta \boldsymbol{x}^k\right\|_2^2)$. Para manter as aproximações realizadas em (3) e (4) o erro de truncamento não pode ser muito grande, isso é controlado adicionando mais uma restrição que restringe Δx^k entre 0 e 1, podendo ser escrita da forma:

$$\left\|\Delta \boldsymbol{x}^k\right\|_1 \leq \beta N_d \tag{6}$$

Isso significa que o número de elementos que vai de sólido para vazio e vice-versa em cada iteração é restrito a uma fração β do número total de variáveis de projeto. O uso de valores pequenos de β garante que o número Δx^k permaneça baixo em cada iteração, mantendo assim o erro de truncamento pequeno. Essa restrição evita que o otimzador realize mudanças muito grandes na estrutura, o que poderia comprometer o projeto.

Formulamos um problema de otimização que pode ser dividido em subproblemas e estes podem ser resolvidos com programação linear inteira. Observe que o problema formulado em 2.1 é um problema binário, para resolvê-lo vamos precisar adotar um modelo de interpolação, neste trabalho vamos adotar o SIMP, de acordo com o método desenvolvido por Raghavendra Sivapuram, Renato Picelli, Yoon G. H. e Yi Bing.

$$E_j = (1 - {p^p}_j) \cdot E_{min} + {p^p}_j \cdot E_s \tag{7}$$

Em que E_j é o módulo de elasticidade do elemento j, p é um fator de penalidade, E_{min} é um número pequeno, da ordem de $10^{-9} \cdot E_s$, empregado a fim de evitar singularidades na análise, pode ser compreendido como o módulo elástico do elemento vazio ($x_j = 0$) e E_s é o módulo elástico do material sólido. Aplicando essa interpolação temos que a formulação em 2.2 pode ser escrita como

$$\begin{aligned} \underset{\rho}{\text{Minimizar}} \quad & f(\rho) \\ \text{Sujeito à} \quad & g_i(\boldsymbol{\rho}) \leq \overline{g}_i,\ i \in [1, N_g] \\ & \rho_{jk} \in \{0,1\} \forall j \in [1, N_e], k \in [1, N_m] \end{aligned}$$

Onde ρ é a matriz de variáveis de projeto de tamanho $N_e \times N_m$. Onde se $\rho_{jk} = 1$ o elemento é sólido e se $\rho_{jk} = 0$ o elemento é vazio.

2.2. Solucionador de Programação Linear Inteira (ILP)

Um problema de programação linear inteira é um programa LP em que o conjunto de soluções possíveis é composto apenas por inteiros. Uma ótima forma de resolver esse tipo de problema é utilizando algoritmos *branch-and-bround*. Em geral, o *branch-and-bround* começa resolvendo o problema LP, sem adicionar restrições de números inteiros, e após isso vão sendo adicionadas restrições conforme o algoritmo resolve os problemas LP, forçando o otimizador a produzir soluções inteiras. No nosso caso, o otimizador deve gerar como output densidades $\rho_{jk} \in \{0,1\}$. Para resolver esses problemas, estamos utilizando a função `intlinprog` do MATLAB, entretanto é possível utilizar pacotes MATLAB de outros otimizadores comerciais, como a função `cplexmilp` do CPLEX - IBM. Para entender a resposta do sistema a mudanças nos parâmetros e garantir que a solução otimizada seja robusta e eficaz em diferentes condições, precisamos realizar agora a análise de sensibilidade. Matematicamente, a sensibilidade é o gradiente da função objetivo e da função de restrição.

$$C(x) = \frac{1}{2} F^T u K(x) u = F g_i = \frac{V(x)}{V_0} \tag{8}$$

$$\frac{dC(x)}{dx_j} = \frac{1}{2} \frac{d(u^T K u)}{dx_j} = u^T K \frac{du}{dx_j} + \frac{1}{2} u^T \frac{dK}{dx_j} u \tag{9}$$

Calculando $\frac{du}{dx_j}$, considerando que o carregamento aplicado é cons-

tante para qualquer variável de projeto x_j, i.e, $\frac{dF}{dx_j} = 0$

$$K(x)u = F \Rightarrow \frac{dF}{dx_j} = \frac{d(Ku)}{dx_j} = 0 \Rightarrow \frac{du}{dx_j} = -K^{-1}\frac{d(Ku)}{dx_j}u \quad (10)$$

$$\therefore \frac{dC(x)}{dx_j} = -\frac{1}{2}u^T\frac{dK}{dx_j}u \quad (11)$$

Para calcular $\frac{dK}{dx_j}$, precisaremos adotar um modelo de material, nesse caso adotamos a SIMP modificada (Solid Isotropic Material Penalization). Nesse trabalho é assumido a formulação de tensões planas, para um material isotrópico, a matriz constitutiva é a seguinte:

$$D = \frac{E(x)}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{(1-\nu)}{2} \end{bmatrix} \quad (12)$$

Utilizando SIMP modificada (Sigmund, 2007)

$$E(x) = E_{min} + {x_j}^p(E_0 - E_{min}) \Rightarrow \frac{dE(x)}{dx_j} = p x_j^{p-1}(E_0 - E_{min}) \quad (13)$$

Calculando $\frac{dK}{dx_j}$:

$$\frac{dK}{dx_j} = \frac{d}{dx_j}(t\int_A B^T DB\, dA) = p{x_j}^{p-1}k_j \quad (14)$$

One B é a matriz da força de corpo em coordenadas nodais, esse resultado pode ser visto em qualquer referência em Elementos Finitos.

Portanto, temos que

$$\frac{dC(x)}{dx_j} = -\frac{1}{2}p{x_j}^{p-1}{u_j}^T k_j u_j \quad (15)$$

Agora precisamos calcular a sensibilidade da função de restrição, em que $g(x) = V(x) = x_j V_j$. Então

$$\frac{dg(x)}{dx_j} = \frac{1}{V_0}\frac{dV(x)}{dx_j} = \frac{V_j}{V_0} \quad (16)$$

Dessa forma obtemos as expressões para as sensibilidades. Isso pode ser implementado utilizando diferenças finitas ou pelo método adjunto (Haftka e Gürdal 1991).

2.3. Filtragem

A filtragem numérica é empregada com finalidade de gerar soluções independentes da malha. No caso de variáveis binárias, faz mais sentido utilizarmos a filtragem das sensibilidades em vez de aplicar-se a filtragem de densidades, uma vez que este está restrito apenas ao conjunto $0, 1$. O campo de sensibilidades filtrado é descrito como

$$\widetilde{\frac{\partial f}{\partial x_j}} = \frac{1}{\sum_{m\in N_m} H_{jm}}\sum_{m\in N_m} H_{jm}\frac{\partial f}{\partial x_m} \quad (17)$$

Em que $H_{jm} = \texttt{max}(0, r_{\min} - \texttt{dist}(\boldsymbol{x}_e, \boldsymbol{x}_m))$, r_{min} é o raio do filtro, N_m é o conjunto de elementos m onde a distância centro a centro do elemento m é menor que o raio do filtro.

3. Otimização de estruturas tridimensionais

Para abranger situações tridimensionais, algumas alterações precisaram ser feitas no código original do artigo educacional citado em 3.2.

- Realizar a análise de elementos finitos separadamente para estruturas tridimensionais;
- Trocar o filtro por um filtro tridimensional, nesse caso r_{min} é o raio da esfera do filtro;
- Recalcular a derivada da matriz de rigidez, uma vez que agora trata-se de um volume e a matriz constitutiva também muda.

Veja que isso é possível pois a convergência é independente da malha, portanto, para trabalhar com estruturas tridimensionais não é preciso realizar mudanças substanciais no algoritmo e na formulação matemática, deve ser feito apenas alguns ajustes nos loops e no filtro.

Neste trabalho, o filtro utilizado foi o empregado no artigo *"An efficient 3D topology optimization code written in Matlab"* (Kai Liu,Andrés Tovar, 2014), este é um filtro de densidades, como estamos buscando filtrar as sensibilidades, alterações foram feitas para que se adeque à implementação do nosso estudo.

Para casos tridimensionais a derivada da matriz de rigidez pode ser escrita como

$$\frac{dK}{dx_j} = \frac{d}{dx_j}(\int_V B^T DB\, dV) = p{x_j}^{p-1}k_j \quad (18)$$

4. Implentação de forças de corpo

Para iniciar o processo de incorporar o peso próprio nós precisamos realizar a análise de elementos finitos incorporando um termo relativo à uma força de corpo $\vec{b}$. De forma que a força de corpo elementar b_e é escrita como:

$$\vec{b_e} = \int_\Omega N^T b d\Omega \quad (19)$$

O que resulta em

$$\vec{b_e} = Vg\rho \cdot \{0, \frac{-1}{8}, 0, 0, \frac{-1}{8}, 0, 0, \frac{-1}{8}, 0\} \quad (20)$$

Depois da força de corpo referente ao peso próprio ter sido incorporada na análise de elementos finitos, precisamos realizar o cálculo da sensibilidade elementar, pois para obter o resultado das equações 7 e 8 , utilizamos que $\frac{dF}{dt} = 0$, o que não é mais verdade, nesse caso. Por isso, precisamos refazer esse cálculo e implementá-lo. Utilizando o método adjunto para calcular a sensibilidade, temos:

$$C(x) = \frac{1}{2}F^T u + \lambda^T(Ku - F) \Rightarrow$$
$$\Rightarrow \frac{dC(x)}{dx_j} = \frac{1}{2}\frac{dF^T}{dx_j}u + \frac{1}{2}F^T\frac{du}{dx_j} + \lambda^T\left(\frac{dK}{dx_j}u + K\frac{du}{dx_j} - \frac{dF}{dx_j}\right) \quad (21)$$

Agrupando $\frac{du}{dx_i}$

$$\frac{dC(x)}{dx_j} = (\frac{1}{2}F^T + \lambda^T K)\frac{du}{dx_j} + \frac{dF^T}{dx_j}u + \lambda^T\frac{dF}{dx_j} \quad (22)$$

Para o caso estático $\frac{du}{dx_j} = 0 \Leftrightarrow \frac{1}{2}F^T + \lambda^T K = 0 \Leftrightarrow \lambda = -\frac{1}{2}u$. Portanto, temos:

$$\frac{dC(x)}{dx_j} = \frac{dF^T}{dx_j}u - \frac{1}{2}u^T\frac{dK}{x_j}u \quad (23)$$

Utilizando os resultados da Equação 13 temos o que segue

$$\frac{dC(x)}{dx_j} = b_j^T u_j - \frac{1}{2}p{x_j}^{p-1}u_j^T K_j u_j \quad (24)$$

Contudo, é importante observar que o vetor $\vec{b}$ precisa ser atualizado a cada iteração. O otimizador buscará uma solução ótima para minimizar a força recebida pelo corpo, removendo massa da estrutura até atingir uma solução trivial. No entanto, essa solução não é adequada para nossos objetivos. Portanto, precisamos impor uma restrição de volume, garantindo que o otimizador atualize a estrutura até alcançar

um volume mínimo V_{min}, evitando assim a remoção completa do material do corpo.

5. Resultados

Primeiramente, vamos estudar o caso de uma viga bi-apoiada, sujeita unicamente ao peso-próprio, causado pela ação da gravidade. Aqui estamos usando como geometria uma seção transversal quadrada de 1m x 1m e comprimento da viga de 5m, mostrada na Figura 1.

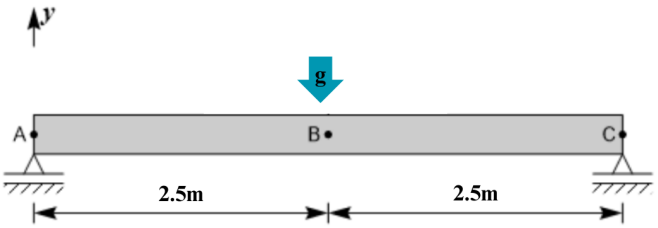

Figura 1. Caso 1: Viga bi-apoiada.

Como parâmetros de otimização, temos que $V_{min} = 0.3V_{inicial}$, o raio do filtro $r_{min} = 0.6m$, $\epsilon = 0.6\%$, $\beta = 1\%$, o módulo de elasticidade dos elementos $x_j = 0$ ("vazio") $E_{min} = 10^{-9}$MPa, o fator de penalidade $p = 4$. Como parâmetros físicos temos a densidade $\rho = 1\frac{kg}{m^3}$, o módulo de elasticidade do material $E_0 = 1$MPa, o Coeficiente de Poisson $\nu = 0.3$ e aceleração da gravidade $g = 9.81\frac{m}{s^2}$. Como parâmetros de malha temos que $nelx = 70$, $nely = 25$ e $nelz = 15$.

O resultado da topologia final, utlizando os parâmetros citados é mostrado na Figura 4.

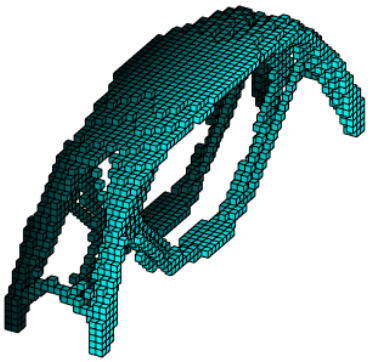

Figura 2. Caso 1: Viga bi-apoiada pós otimização sujeita apenas ao próprio-peso.

Utilizando os mesmos parâmetros implemntados para o Caso 1, agora iremos realizar a otimização da mesma viga da Figura 1 mas desconsiderando o peso-próprio e a mesma sujeita apenas à uma força externa distribuída, como mostrado na Figura 5.

Para essa situação obtemos a topologia otimizada da Figura 6 e os gráficos das Figuras 7 e 8.

Note que a otimização do Caso 1 é muito mais suave e mais eficiente em relação à topologia obtida no Caso 2, mostrando a importância do desenvolvimento de novas formulações para otimizar os métodos.

6. Conclusão

Com isso, concluímos que a solução para o problema inicial, que consistia no otimizador fornecer uma resposta nula quando o peso-próprio é implementado, envolve recalcular as sensibilidades e impor

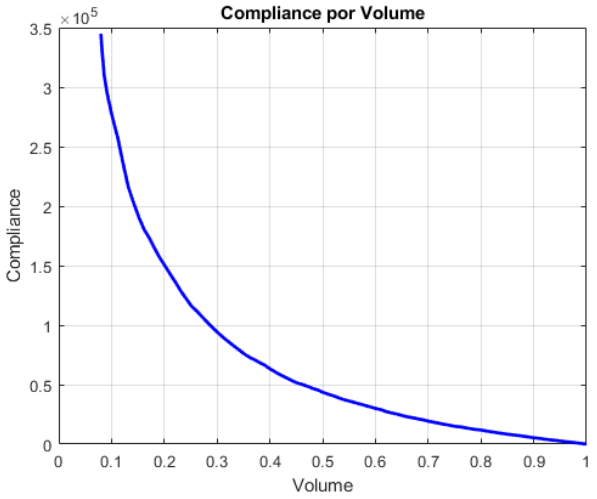

Figura 3. Caso 1: Compliance normalizada vs Volume normalizado.

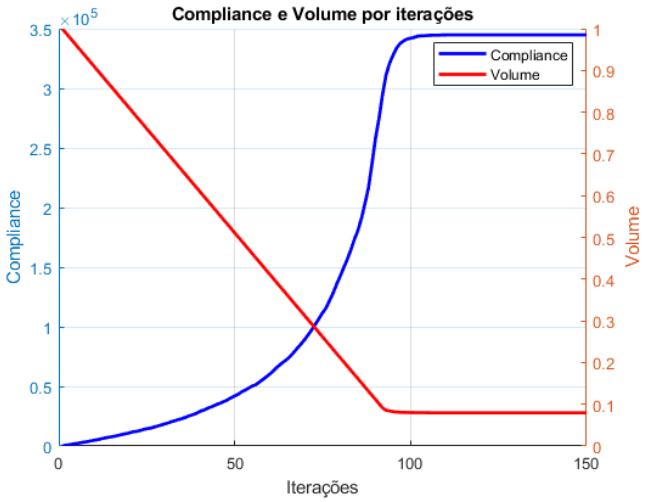

Figura 4. Caso 1: Compliance normalizada e Volume mormalizado vs Número de Iterações.

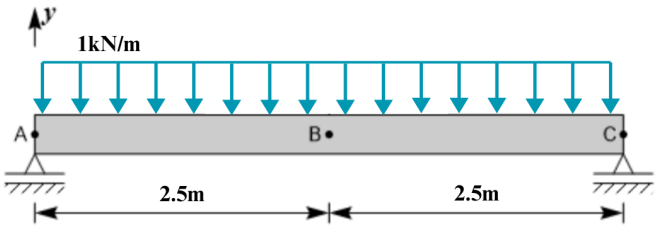

Figura 5. Caso 2: Viga bi-apoiada sujeita à uma força externa concentrada.

uma restrição de volume. Dessa forma, o otimizador remove material até alcançar um valor mínimo V_{min} pré-definido.

Esse resultado pode facilitar a otimização de estruturas grandes, como prédios, pontes e torres, de maneira mais eficiente. Portanto, observamos que o desenvolvimento de novas abordagens na implementação das ferramentas de otimização topológica possibilita estruturas ainda mais eficientes e econômicas, garantindo segurança aos usuários e reduzindo o consumo de recursos.

7. Bibliografia

1. Andreassen E, Clausen A, Schevenels M, Lazarov BS, Sigmund O (2011) Efficient topology optimization in MATLAB using 88

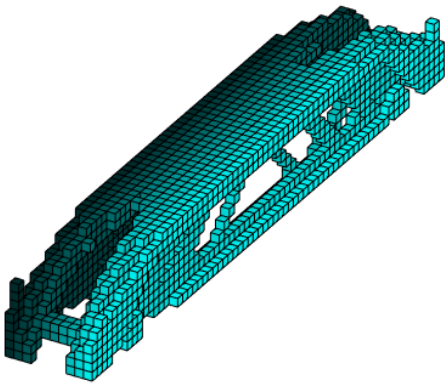

Figura 6. Caso 2: Viga bi-apoiada pós otimização sujeita à um carregamento externo distribuído.

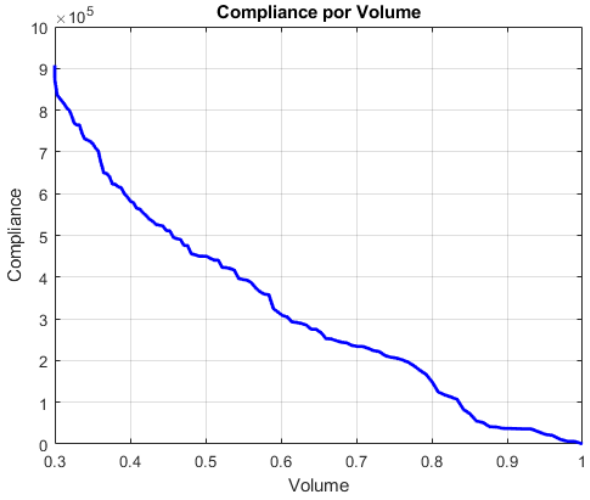

Figura 7. Caso 2: Compliance normalizada vs Volume normalizado.

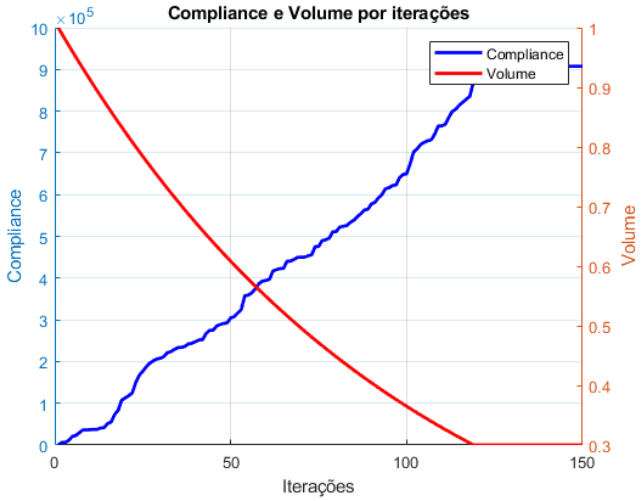

Figura 8. Caso 2: Compliance normalizada e Volume mormalizado vs Número de Iterações.

lines of code. Struct Multidiscip Optim 43:1–16

2. Beckers M (1999) Topology optimization using a dual method with discrete variables. Struct Multidiscip Optim 17:14–24
3. Bendsoe, M.P.(2005). Optimization of structural topology, shape, and material. Springer-Verlag, Heidelberg.
4. Haftka RT. Gürdal, Zafer (1991) Elements of Structural Optimization. Springer Dordrecht. https://doi.org/10.1007/978-94-011-2550-5
5. Haftka RT, Grandhi RV (1986) Structural shape optimization—a survey. Comput Methods Appl Mech Eng 57:91–106. https://doi.org/10.1016/0045-7825(86)90072-1
6. Huang, X. and Xie, Y. M. (2011). Evolutionary topology optimization of continuum structures including design-dependent self-weight loads. Finite Elements in Analysis and Design, 47:942–948.
7. Huang, X. and Xie, Y. M. (2007). Convergent and mesh-independent solutions for the bidirectional evolutionary structural optimization method. Finite Elements in Analysis and Design, 43:1039–1049.
8. Liu, K., Tovar, A (2014). An efficient 3D topology optimization code written in Matlab. Struct Multidisc Optim 50, 1175–1196; https://doi.org/10.1007/s00158-014-1107-x
9. Picelli, R., Sivapuram, R. & Xie, Y.M. (2021). A 101-line MATLAB code for topology optimization using binary variables and integer programming. Struct Multidisc Optim 63, 935–954. https://doi.org/10.1007/s00158-020-02719-9
10. Picelli, R. Sivapuram, Raghavendra. (2020). Solving topology optimization with 0,1 design variables and mathematical programming: the TOBS method.
11. Raghavendra Sivapuram, Renato Picelli, Gil Ho Yoon, Bing Yi (2021). On the design of multimaterial structural topologies using Integer Programming, Computer Methods in Applied Mechanics and Engineering, Volume 384. https://doi.org/10.1016/j.cma.2021.114000
12. Seitz, K. F. and Grabe, J. (2016). Three-dimensional topology optimization for geotechnical foundations in granular soil. Computers and Geotechnics, 80:41–48.
13. Sigmund O (2001a) A 99 line topology optimization code written in Matlab. Struct Multidiscip Optim 21:120–127
14. Sivapuram R, Picelli R and Xie YM (2018). Topology optimization of binary microstructures involving various non-volume constraints. Comput. Mater. Sci., 154:405–425
15. Sivapuram, R., Picelli, R (2020). Topology design of binary structures subjected to design-dependent thermal expansion and fluid pressure loads. Struct Multidisc Optim 61, 1877–1895. https://doi.org/10.1007/s00158-019-02443-z
16. Svanberg K, Werme M (2006) Topology optimization by a neighbourhood search method based on efficient sensitivity calculations. Int J Numer Meth Eng 67(12):1670–1699. https://doi.org/10.1002/NME.1677
17. Xie, Y. and Steven, G (1997). Evolutionary Structural Optimization. Springer Verlag.
18. Zuo ZH, Xie YM (2015) A simple and compact python code for complex 3D topology optimization. Adv Eng Softw 85:1–11