Pontifícia Universidade Católica
do Rio de Janeiro

Vinícius Gama Tavares

Sistema eficiente de otimização topológica estrutural utilizando o método de malha densa de barras

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio.

Orientador: Prof. Waldemar Celes Filho

Rio de Janeiro
Março de 2017

Pontifícia Universidade Católica
do Rio de Janeiro

Vinícius Gama Tavares

Sistema eficiente de otimização topológica estrutural utilizando o método de malha densa de barras

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Waldemar Celes Filho
Orientador
Departamento de Informática – PUC-Rio

Prof. Marcelo Gattass
Departamento de Informática – PUC-Rio

Prof. Ivan Fábio Mota de Menezes
Departamento de Engenharia Mecânica – PUC-Rio

Prof. Luiz Henrique de Figueiredo
IMPA

Prof. Marcio da Silveira Carvalho
Coordenador Setorial do Centro Técnico Científico – PUC-Rio

Rio de Janeiro, 31 de Março de 2017

Vinícius Gama Tavares

Graduou-se em Engenharia da Computação pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), onde trabalhou em projetos no grupo de visualização do Instituto Tecgraf. Em 2015 ingressou no programa de mestrado em Informática na Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio). Durante o mestrado, continou seu trabalho e sua pesquisa no grupo de visualização do Instituto Tecgraf.

Ficha Catalográfica

Tavares, Vinícius Gama

Sistema eficiente de otimização topológica estrutural utilizando o método de malha densa de barras / Vinícius Gama Tavares; orientador: Waldemar Celes Filho. – 2017.

v., 68 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Informática – Teses. 2. Malha densa de barras;. 3. Otimização topológica de barras;. 4. Programação linear.. I. Celes Filho, Waldemar. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Ao meu orientador Waldemar Celes, por toda sua atenção e paciência ao longo de todos os anos como meu professor, sempre confiando em meu potencial.

Ao professor Ivan Menezes, por sua simpatia, me motivando e ajudando com as dúvidas que tive durante a pesquisa.

À minha companheira Raphaela Curvão, por seu carinho, apoio e incentivo durante todo este trabalho, sempre estando ao meu lado.

Aos meus pais, Roberto Tavares e Ana Célia Gama, e à minha irmã, Fernanda Tavares, por todo o suporte que me dão e a dedicação que possuem comigo.

Aos meus colegas do Tecgraf, em especial a Rodrigo Espinha, Leonardo Duarte, Alice Herrera e Vicente Neto, por me ajudarem em várias etapas desta pesquisa.

Ao CNPq, ao Tecgraf e à PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Resumo

Tavares, Vinícius Gama; Celes Filho, Waldemar. **Sistema eficiente de otimização topológica estrutural utilizando o método de malha densa de barras**. Rio de Janeiro, 2017. 68p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Métodos de otimização topológica estrutural visam obter a melhor distribuição de material dentro de um dado domínio, sujeito a carga, condições de contorno e restrições de projeto, de forma a minimizar alguma medida especificada. A otimização topológica estrutural pode ser dividida em dois tipos: contínua e discreta, sendo a forma discreta o foco da pesquisa desta dissertação. O objetivo deste trabalho é a criação de um sistema para realizar todos os passos dessa otimização, visando a resolução de problemas com grandes dimensões. Para realizar esse tipo de otimização, é necessária a criação de uma malha densa de barras, esta definida como conjunto de nós cobrindo todo o domínio, conectados através de barras, além da especificação dos apoios e das forças aplicadas. Este trabalho propõe um novo método para geração da malha densa de barras, utilizando como entrada somente o contorno do domínio que se deseja otimizar, contrapondo com métodos que necessitam de um domínio já discretizado, como uma malha de poliedros. Com a malha gerada, este trabalho implementou a otimização topológica, sendo necessário resolver um problema de programação linear. Toda a parte de otimização foi realizada dentro do framework TopSim, tendo implementado o método dos pontos interiores para a resolução da programação linear. Os resultados apresentados possuem boa qualidade, tanto na geração quanto na otimização, para casos 2D e 3D, tratando casos com mais de 68 milhões de barras.

Palavras-chave

Malha densa de barras; Otimização topológica de barras; Programação linear.

Abstract

Tavares, Vinícius Gama; Celes Filho, Waldemar (Advisor). **Efficient structural topology optimization system using the ground structure method**. Rio de Janeiro, 2017. 68p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Structural topology optimization methods are used to find the optimal material distribution within a given domain, subject to loading, boundary conditions and design constraints, in order to minimize some specified measure. Structural topology optimization can be divided into two types: continuum and discrete, with the discrete type being the research focus of this dissertation. The goal of this work is the creation of a system to achieve all the steps of this optimization process, aiming problems with large dimensions. In order to perform the optimization, it is necessary create a ground structure, defined as a set of nodes covering the entire domain, connected by bars, with the supports and the applied loads. This work proposes a new method for the ground structure generation, using as input only the domain boundary, in contrast with methods that require a domain already discretized, such as a polyhedron mesh. With the generated mesh, this work has implemented the topological optimization, needing to solve a linear programming problem. All the optimization part was performed within the TopSim framework, implementing the interior point method for the linear programming resolution. The results presented have good quality, both in generation and optimization, for 2D and 3D cases, considering cases with more than 68 million bars.

Keywords

Ground structure method; Topology optimization of trusses; Linear programming.

Sumário

Lista de figuras

Lista de tabelas

1
Introdução

A otimização topológica estrutural visa a redução de custos para a produção e a redução e/ou balanceamento dos níveis de tensão nos componentes de uma determinada estrutura. Para isso ser possível, técnicas de otimização estrutural conseguem, por exemplo, identificar a estrutura de treliça ótima de um determinado projeto definido por suas cargas e restrições.

Treliças são sistemas constituídos por elementos unidimensionais, unidos entre si por articulações e sujeitas apenas a cargas aplicadas nas articulações (nós). Assim, os elementos (barras) ficam exclusivamente sujeitos a esforços normais, de tração ou compressão. Elas são utilizadas como estruturas de coberturas em edifícios estruturais, em torres de transmissão de energia, pontes e em diversas outras construções.

Este trabalho foca no processo de otimização topológica estrutural discreta que, através de um número finito e discreto de elementos, consegue fornecer uma aproximação da solução analítica encontrada para as estruturas de Michell (1). Esse método é conhecido como *ground sctructure method* (*GSM*) (2) e, para sua execução, é necessária a geração de uma malha densa de barras inicial. Dorn (2) define uma malha densa de barras como uma grade que contem todas as juntas estruturais (nós), apoios, forças aplicadas e é conectada por todos seus potenciais membros (barras) (3).

Para remover as barras desnecessárias da malha e encontrar a estrutura de treliça ótima, o problema de otimização, se respeitadas algumas premissas e utilizada sua formulação plástica, pode ser visto como um problema de programação linear (5). Dependendo do resultado obtido, pode-se fazer necessário um processo de filtragem na estrutura para se chegar a um modelo realmente possível de ser construído ou manufaturado. Com essa filtragem, é possível gerar modelos compatíveis com impressoras 3D, economizando material da impressão e criando estruturas robustas. A Figura 1.1 mostra o caso da otimização de um gancho, com as condições de contorno, a malha densa de barras e a treliça ótima.

As principais contribuições desta dissertação são:

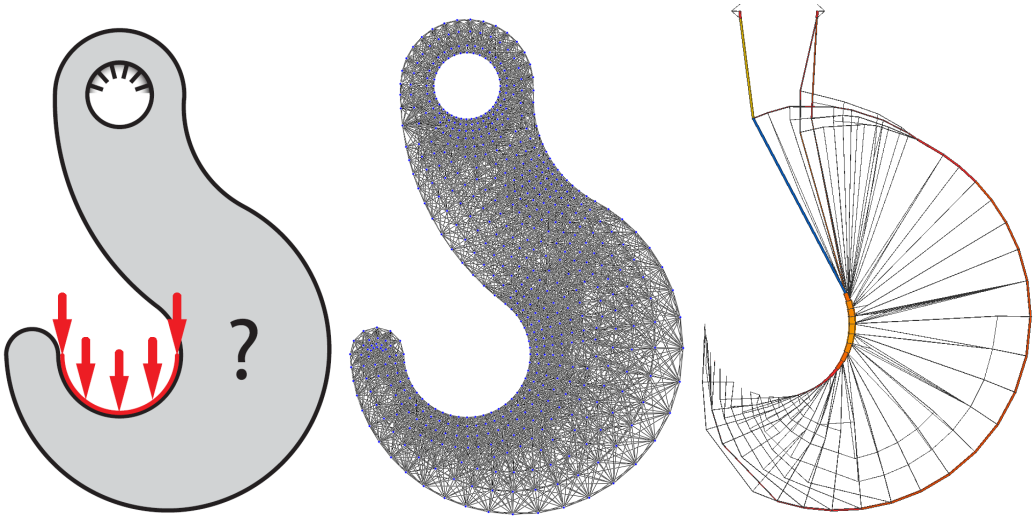
– Uma nova forma de geração de malha densa de barras, na qual a malha é gerada a partir de uma descrição do contorno do domínio, contrapondo

com as técnicas que necessitam de um modelo discretizado.

– A implementação de um método de programação linear dentro do *framework* de simulação proposto por Duarte (4), com a finalidade de prover suporte a problemas de larga escala

Os resultados obtidos são verificados através de sua visualização e comparação com modelos em que já se conhece o resultado esperado.

O trabalho está organizado da seguinte forma: no Capítulo 2 são apresentados trabalhos relacionados a geração da malha densa de barras e sua otimização; o Capítulo 3 explica os conceitos da malha densa de barras e o algoritmo proposto por esta dissertação para sua geração. O Capítulo 4 apresenta a otimização topológica estrutural de barras e como ela foi implementada neste trabalho. O Capítulo 5 exibe os resultados obtidos com a otimização, a qualidade obtida e seu desempenho. Por fim, no Capítulo 6 apresenta-se as considerações finais desta dissertação e possíveis trabalhos futuros.



(a) (b) (c)

Figura 1.1: Processo de otimização de uma estrutura na forma de gancho: (a) Domínio, forças aplicadas e condições de contorno. (b) Exemplo de uma malha densa de barra gerada. (c) Resultado da otimização.

2
Trabalhos Relacionados

A geração de malhas densas no software *GRAND - GRound structure ANalysis and Design* (5, 6), será usada como base para o desenvolvimento deste trabalho. O GRAND realiza tanto a geração da malha densa de barras quanto a otimização estrutural. Ele foi desenvolvido em MATLAB e foi feito para domínios arbitrários, isto é, domínios que podem apresentar contornos curvos e furos internos.

Para gerar a estrutura densa de barras, o GRAND espera receber uma malha de polígonos, que pode ser regular ou um diagrama de Voronoi, além de "zonas de restrição", que podem ser retângulos, círculos, linhas ou polígonos convexos. Essas zonas são regiões que não permitem a existência de uma barra nos seus interiores.

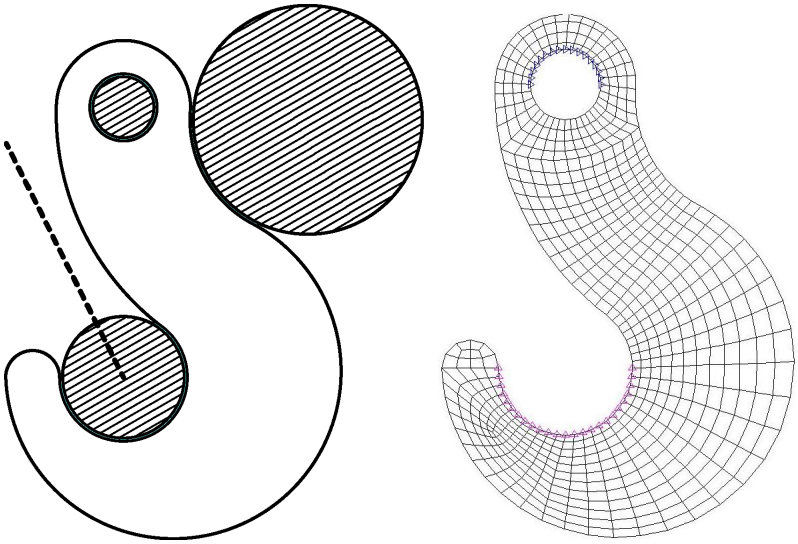
Apesar de produzir malhas densas de boa qualidade, a metodologia empregada no GRAND apresenta algumas limitações. A principal limitação está na exigência de um modelo discreto (malha de polígonos/poliedros) inicial. Obter um modelo discreto é, em geral, um problema mais complexo do que uma malha densa de barras, em especial para domínios 3D complexos.

Outras desvantagens são: necessidade de uma estrutura topológica para representação das adjacências da malha inicial, necessidade de identificação explícita das "zonas de conflito" e teste exaustivo de intersecção com estas zonas, além da dificuldade de aplicação em domínios 3D com fronteiras complexas. A Figura 2.1 exemplifica essas "zonas de restrição" e a malha poligonal usada para o caso do gancho.

Zhang e outros (7) propuseram outro método para geração da malha densa de barras, porém também dependente de uma malha de polígonos. Duas abordagens são definidas nesse trabalho: a *Macroelement* e a *Macropatch*.

Na abordagem *Macroelement*, são inseridos nós igualmente espaçados em cada aresta de cada elemento da malha de polígonos. As barras são criadas somente entre os nós de cada elemento. Já na abordagem *Macropatch*, cada elemento é subdividido em subelementos e, então, as barras são criadas entre os nós desses novos subelementos. A Figura 2.2 exibe as diferenças entre cada abordagem.

O método proposto nesta dissertação utiliza uma simples estrutura de



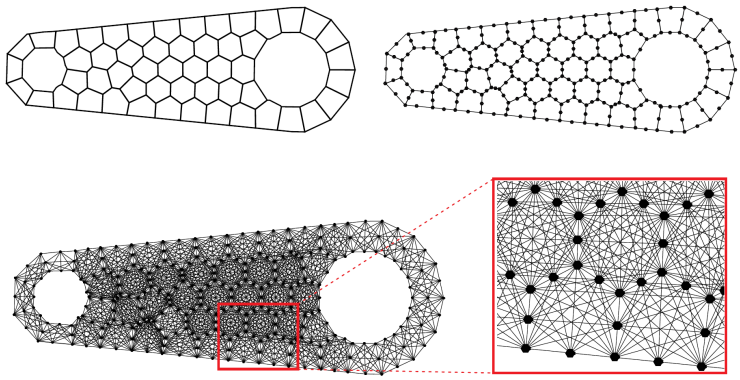
(a) Zonas de restrição

(b) Malha de poligonais, forças aplicadas em vermelho e condições de contorno em azul

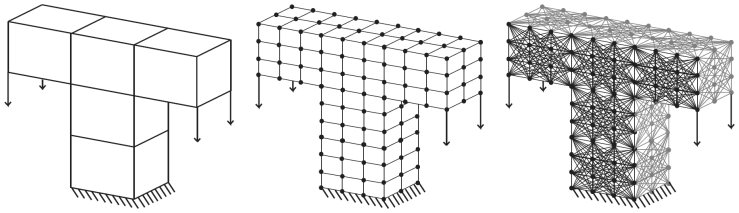
Figura 2.1: Exemplo de entrada utilizada pelo GRAND

grade regular sendo aplicável em domínios 2D e 3D, sem a necessidade da existência de um modelo discretizado do domínio de interesse. A partir da descrição da fronteira do domínio, o método proposto gera a malha densa de barras.

Na área da programação linear para realizar a otimização, Cubas Rodriguez (8) implementou em sua dissertação de mestrado o Método de Pontos Interiores utilizando a *GPU*, através de um *solver* iterativo elemento por elemento, sendo apresentadas resoluções de problemas com até 30 milhões de barras. A Figura 2.3 exibe o resultado da otimização do gancho obtido com uma malha densa de barra composta por 5 milhões de barras.



(a) Abordagem *Macroelement*



(b) Abordagem *Macropatch*

Figura 2.2: Diferença na discretização entre as duas abordagens proposta por Zhang e outros (7)

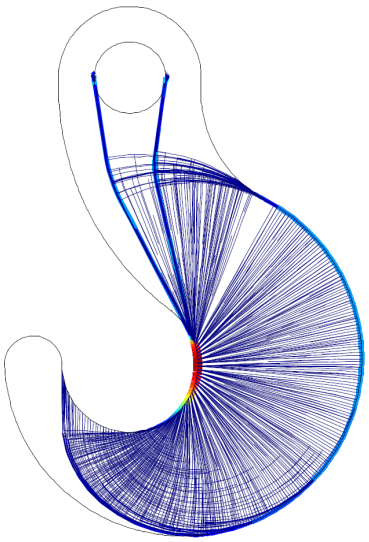
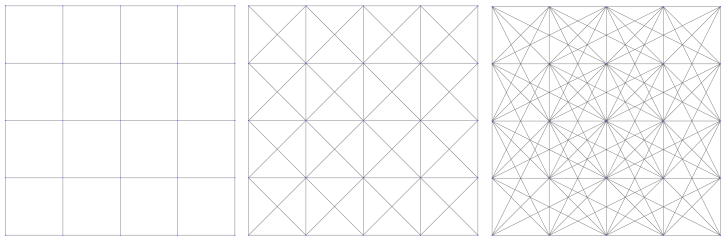


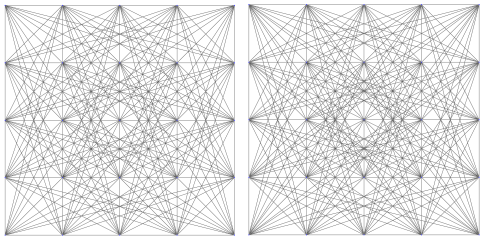
Figura 2.3: Resultado da otimização proposta por Cubas Rodriguez (8) com 5M de barras

3
Geração da Malha Densa de Barras

Uma malha densa de barras é constituída de nós que discretizam o domínio e esses nós são conectados por meio de barras (7). Teoricamente, há uma conexão direta entre quaisquer dois nós, sempre respeitando os contornos e as restrições do domínio. Esse caso, apesar de ser o ideal, pode levar a um custo computacional alto, devido ao grande número de barras presentes na malha. Para reduzir esse custo, regula-se a densidade de barras a serem geradas no modelo. Isto é, escolhe-se o quão densa será a malha gerada. Como exemplo, o trabalho GRAND define a densidade da malha a partir de um conceito topológico, chamado *nível de conectividade*. Nele, cria-se barras usando o conceito de vizinhança dos nós, isto é, se dois nós compartilham o mesmo elemento, eles são considerados vizinhos. Com isso, o nível de conectividade 1 irá gerar barras entre todos os nós vizinhos, o nível 2 irá englobar o nível 1 mais as barras entre os vizinhos dos vizinhos e assim por diante. A Figura 3.1 exibe exemplos de diferentes níveis de conectividade e seu impacto na malha gerada.



(a) Modelo base para o GRAND (b) Nível 1 (c) Nível 2



(d) Nível 3 (e) Totalmente conectado

Figura 3.1: Densidade da malha gerada para diferentes níveis de conectividade

Na Figura 3.2 é feita a comparação do resultado final da otimização para

diferentes níveis de conectividade para o modelo Michell (1). Nota-se que a Figura 3.2(d), com nível de conectividade 5 é semelhante as Figuras 3.2(e) e 3.2(f), que são idênticas e possuem nível 7 e 9 respectivamente. O número de barras em cada caso está presente na Tabela 3.1. Muitas barras no processo de otimização são descartadas (7, 13), sendo que, quando barras maiores podem ser representadas por barras menores, as maiores tendem a ser removidas (5), o que explica a Figura 3.2(f) ter mais de 60000 barras do que a Figura 3.2(e) e, mesmo assim, terem resultados iguais. É muito difícil definir um nível de conectividade ideal para todos os casos, sendo necessário especificá-lo para cada problema, levando em consideração o nível de detalhamento esperado e o custo computacional que isso irá acarretar.

Caso	Número de Barras
Figura 3.2(b)	3947
Figura 3.2(c)	28217
Figura 3.2(d)	76034
Figura 3.2(e)	137217
Figura 3.2(f)	199807

Tabela 3.1: Número de barras na malha densa para diferentes níveis de conectividade

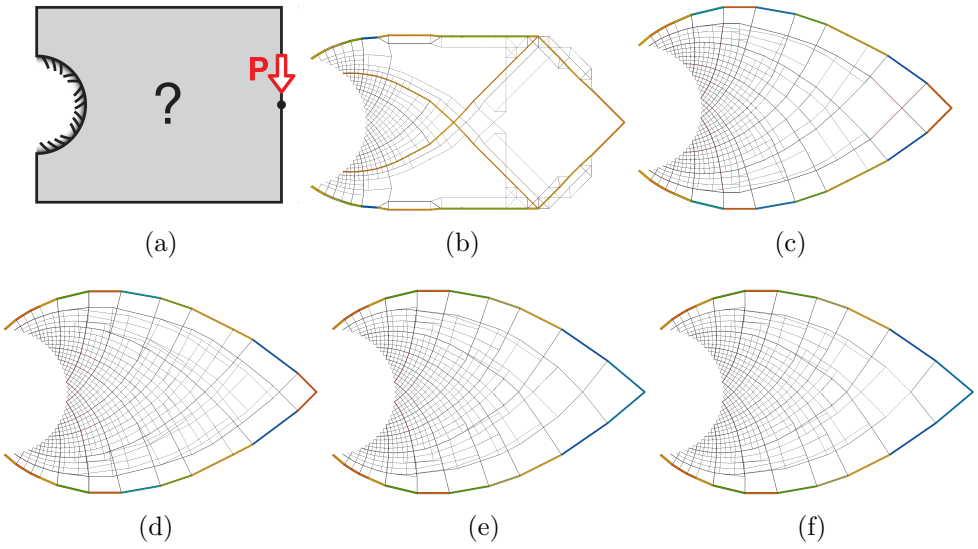
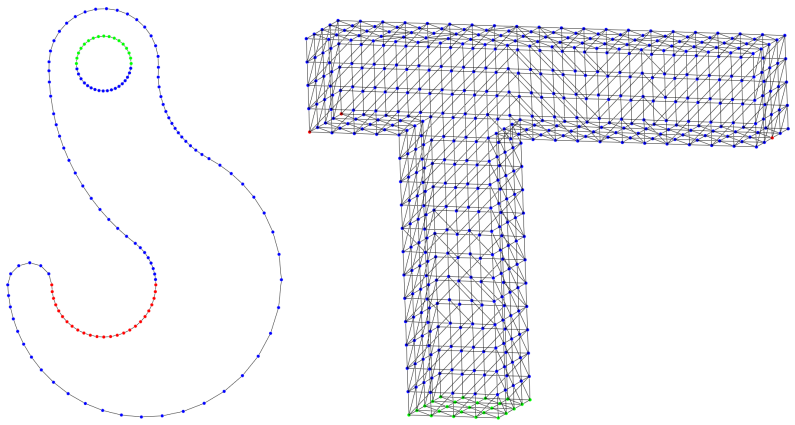


Figura 3.2: Variação no resultado da otimização devido à diferentes níveis de conectividade no caso Michell: (a) Domínio, forças aplicadas e condições de contorno. (b) Nível de conectividade = 1. (c) Nível de conectividade = 3. (d) Nível de conectividade = 5. (e) Nível de conectividade = 7. (f) Nível de conectividade = 9

Outro ponto importante na malha densa de barras é a proibição da existência de barras colineares na malha final. Isso é necessário pois evita múltiplas soluções possíveis. Além disso, como será usada a programação linear

neste trabalho, barras colineares geram matrizes singulares, impossibilitando sua otimização. Para isso, é preciso definir uma tolerância máxima do ângulo entre as barras, para assim se evitar a colinearidade. Além disso, se existirem barras colineares, a de maior tamanho deve ser removida, devido ao que foi discutido anteriormente com os resultados da Figura 3.2.

Para gerar a malha densa de barras, este trabalho espera como entrada uma descrição da fronteira do domínio, as restrições do problema e quais forças são aplicadas. No caso 2D, a descrição é feita através de segmentos de reta e, para o caso 3D, são utilizados triângulos. A Figura 3.3 mostra exemplos da entrada esperada. Os nós especificados na entrada servirão de "sementes" para discretizar o domínio, com o tamanho das arestas e seus ângulos servindo como informações para a criação dos nós internos e das barras. Todo o processo de geração da malha densa de barra proposto é explicado a seguir.



(a) Caso 2D (b) Caso 3D

Figura 3.3: Exemplos de entrada esperada para este trabalho. Os nós verdes são as restrições do problema e os nós vermelhos são as forças aplicadas

3.1 Criação da grade regular

O passo inicial da geração é a criação de uma grade regular na qual todo o domínio ficará contido. A escolha de usar uma grade regular deve-se pela facilidade que existe para descobrir em qual célula um ponto qualquer se encontra, visando acelerar o processo como um todo da geração de barras.

A dimensão escolhida para a célula da grade irá influenciar no resultado final da geração de malha, já que quanto mais refinada for a grade, maior será a precisão no cálculo do campo de distância que será realizado (e explicado mais à frente), porém isso também aumentará o custo computacional da geração.

Para facilitar essa escolha, a dimensão da célula é definida proporcionalmente à média dos tamanhos das arestas da entrada, tendo o lado l da

célula valor igual a $l = \frac{\sum_{i=1}^{n} dist_i}{n}\tau$, em que n é o número de arestas da malha de entrada, $dist_i$ é a distância entre dois nós conectados e τ é o fator de multiplicação escolhido para o problema.

3.2 Classificação das Células

Com a grade criada, se faz necessário a classificação de suas células, sendo criado quatro tipos: Indefinidas (i.e., ainda não foi classificada); Fora do Domínio; Dentro do Domínio; e na Fronteira do Domínio. Inicialmente, todas as células são classificadas como indefinidas. Para classificá-las, este trabalho usou a técnica de preenchimento *Flood fill* (14). Primeiro, as células do contorno são classificadas e, com a fronteira já delimitada, as células restantes são então classificadas. Para descobrir as células da fronteira, foi necessária a implementação de algoritmos distintos para os casos 2D e 3D.

3.2.1 Classificação da Fronteira em 2D

Para encontrar todo o contorno, é preciso descobrir quais células os segmentos de reta do domínio interceptam. Para isso, foi implementada uma técnica de caminhamento pelas células da grade proposta por Amanatides e Woo (15), executando-a para cada aresta.

Como o algoritmo proposto por Amanatides e Woo é para ser utilizado em algoritmos de traçados de raio, a aresta é então escrita na forma paramétrica $\boldsymbol{u} + t\boldsymbol{v}$. A principal ideia desse algoritmo é encontrar os valores de t de tal forma que cada célula seja visitada somente uma vez a cada iteração. Para isso, o algoritmo é dividido em duas partes: inicialização e caminhamento. Na fase de inicialização, o ponto é deslocado para a borda mais próxima da célula ao qual pertence. A partir disso, é realizada a parte do caminhamento, em que se calcula o valor de t de tal forma que seu valor intercepte a borda da próxima célula. A Figura 3.4 ilustra o caminhamento do ponto p_1 até o ponto p_2, em que as células vermelhas são as células visitadas (ou seja, da fronteira) e os pontos azuis são os pontos encontrados pelo algoritmo durante sua execução.

3.2.2 Classificação da Fronteira em 3D

O método usado para 3D é análogo ao caso 2D, sendo necessário descobrir quais são as células que os triângulos interceptam. Com isso, foi implementada a técnica de colisão entre um triângulo e uma caixa proposta por Akenine-Möller (16). O algoritmo apresenta os seguintes passos:

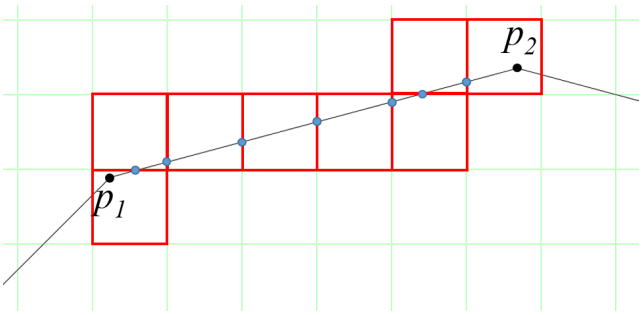


Figura 3.4: Exemplo de execução do caminhamento proposto por Amanatides e Woo (15). As células vermelhas são as células visitadas pelo caminhamento, com os pontos azuis sendo o ponto da borda visitado em cada iteração.

- Calcula-se a caixa envolvente do triângulo que deseja descobrir quais células ele intercepta.
- Com a caixa envolvente, encontra-se quais são as células candidatas a colidirem com o triângulo.
- Para cada célula candidata, são feitos 13 testes para determinar se há ou não a colisão entre o triângulo e a célula.
- Se todos os testes passarem, então o triângulo intercepta a célula e ela é então classificada.

3.2.3 Classificação das Células Restantes

Enquanto as células de fronteira vão sendo classificadas, elas também são adicionadas a um conjunto chamado *conj*. Terminada a classificação de todas as células de contorno, o algoritmo para a classificação das células passa a iterar e remover as células do conjunto *conj*.

Definindo os vizinhos da célula a como sendo todas as células que compartilham um vértice com a, o algoritmo retira do conjunto *conj* a célula *cel* e computa seus vizinhos. Passa-se então a iterar sobre os vizinhos de *cel*. Para cada vizinho c de *cel*, verifica-se sua classificação. Se c ainda não foi classificado, ele é então classificado da seguinte forma: se c for vizinho a uma célula que já foi classificada como dentro ou fora do domínio, então c é desse mesmo tipo, já que o contorno está fechado; se c for vizinho somente a células de fronteiras e/ou células indefinidas, então é necessário que c passe por um teste. Esse teste é definido através da seguinte regra (17):

> Dado um ponto x qualquer, um volume V determinado por uma superfície de contorno S, uma amostragem P da superfície S, um ponto $p \in P$ que é a amostra mais próxima a x, um vetor n_p normal à S em p e orientado para fora de S, temos que:

$$(\boldsymbol{x} - \boldsymbol{p}) \cdot \boldsymbol{n_p} > 0 \Leftrightarrow \boldsymbol{x} \notin V. \tag{3-1}$$

Isto é, essa regra visa determinar em qual parte da superfície formada pelo ponto do contorno e pela normal, o vetor entre os pontos está, podendo assim definir se o ponto x está dentro ou fora do domínio.

Com essa regra, os vértices v_i que compõem c são analisados através da Equação (3-1), comparando-os aos mais próximos nós que pertencem a células de fronteira vizinhas. Então, para todo v_i é encontrado o ponto d da fronteira mais próximo a ele, sendo v_i classificado com a normal relativa à d através da Equação (3-1). Após todos os v_i serem classificados, a célula c é classificada com a mesma classificação de seus vértices.

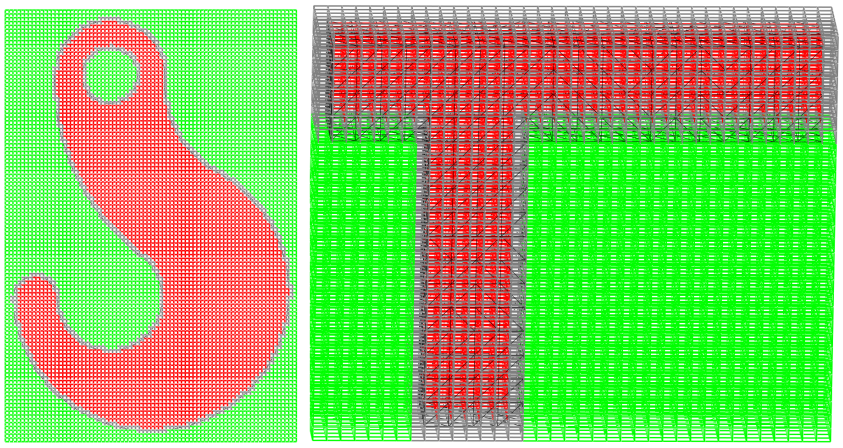
Ao passar pela classificação, somente as células que estão dentro do domínio são colocadas no conjunto. Isso visa acelerar o processo de classificação das células, já que as vizinhas de células fora do domínio também estão fora. Sendo assim, ao final do algoritmo, isto é, quando o conjunto estiver vazio, todas as células indefinidas estão fora do domínio. O Algoritmo 1 exibe o pseudocódigo para esse algoritmo. O resultado dessa classificação pode ser visto na Figura 3.5 para exemplos 2D e 3D, em que as células vermelhas foram classificadas como dentro do domínio, as cinzas como de fronteira e as verdes são das células fora do domínio.

Algoritmo 1 *Flood fill*, usado para a classificação das células que não sejam de fronteira

```
1:  while !conj.vazio() do
2:      cel = conj.proxima_celula()
3:      vizinhos = computa_vizinhos(cel)
4:      for all Celula c em vizinhos do
5:          if c.tipo() == indefinido then
6:              classifica(c)
7:              if c.tipo() == dentro_dominio then
8:                  conj.adiciona(c)
9:              end if
10:         end if
11:     end for
12: end while
```

3.3 Geração dos nós

Para a discretização do domínio, os nós presentes na entrada são usados como "sementes" para a geração dos nós do interior. Para isso, é calculado o campo de distância do domínio, sendo esse campo usado para guiar a direção de propagação de novos nós. Além disso, informações sobre os segmentos de retas



(a) Caso 2D (b) Caso 3D

Figura 3.5: Exemplo de classificação das células, em que as verdes foram classificadas como fora do domínio, as cinzas como na fronteira e as vermelhas como dentro do domínio

aos quais o nó semente pertence são usadas para determinar o espaçamento da propagação.

3.3.1 Campo de Distância

Um campo de distância é uma representação onde, para cada ponto dentro do domínio, é conhecida a distância dele para a fronteira do domínio (18). Além do valor da distância, é possível inferir também outras propriedades, como a direção para o contorno da fronteira, através do gradiente do campo, sendo essa a propriedade usada neste trabalho.

Definindo Σ como o conjunto de pontos do contorno, tem-se que função de distância de um campo contínuo para um ponto $\boldsymbol{p}$ qualquer é:

$$dist_{\Sigma}(\boldsymbol{p}) = \inf_{\boldsymbol{x}\in\Sigma} ||\boldsymbol{x} - \boldsymbol{p}|| \tag{3-2}$$

Sendo a função de distância com sinal definida como:

$$d_s(\boldsymbol{p}) = sgn(\boldsymbol{p}) dist_{\Sigma}(\boldsymbol{p}) \tag{3-3}$$

Em que:

$$sgn(\boldsymbol{p}) = \begin{cases} -1, & \text{se } \boldsymbol{p} \text{ está dentro do domíno} \\ 1, & \text{caso contrário} \end{cases}$$

Uma importante propriedade da função de distância com sinal d é que $||\nabla d|| = 1$ em quase todo o lugar. A exceção são pontos que possuem mais de um ponto da fronteira como mais próximos, como por exemplo o centro de

uma esfera. Pontos nessa região, conhecida como eixo medial, não possuem o gradiente definido (18).

Sendo o gradiente definido no ponto $\boldsymbol{p}$, ele é ortogonal à isolinha (ou isosurperfície no caso 3D) que passa em $\boldsymbol{p}$ (18). Essa informação é utilizada neste trabalho, já que, com o gradiente, é possível andar dentro do domínio seguindo a forma de seu contorno, partindo de sua fronteira em direção ao eixo medial.

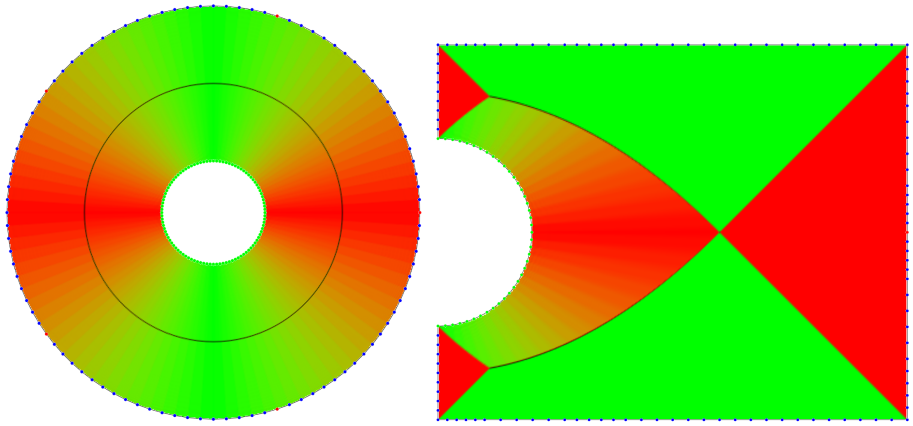
Para este trabalho, é computado o campo de distância discreto, em que os vértices da grade terão um valor aproximado de sua distância para a fronteira. Para isso, inicialmente é calculada a distância dos vértices das células da fronteira para o contorno. Isto é, para o caso 2D, é preciso computar a distância de um ponto para um segmento de reta, enquanto no caso 3D é computada a distância de um ponto para um triângulo.

Computados esses valores, eles são então propagados para o restante dos vértices. Neste trabalho, é minimizada a função distância usando a distância Euclidiana e os valores são propagados através da transformada de distância de vetor, em que cada vértice armazena um vetor com sua distância para fronteira, usando o esquema de varredura, em que a propagação se inicia no vértice 0 da célula de coordenadas $(0, 0)$ (ou $(0, 0, 0)$ no caso 3D), caminhando linha por linha da grade até o seu final. Essa passada é chamada de "passada para frente" (*forward pass*). Depois é executada a "passada para trás" (*backward pass*), em que se parte do final da grade até o seu começo. Podem ser necessárias várias passadas até que o valor de cada vértice esteja correto. Isso ocorre quando tanto a *foward pass* e a *backward pass* não alteram o valor da distância de nenhum vértice. O Algoritmo 2 exibe o pseudocódigo usado neste trabalho para a *forward pass* no caso 2D, em que x e y representam os vértices da grade. A implementação para o caso 3D é análoga, sendo necessário caminhar também pelo eixo z.

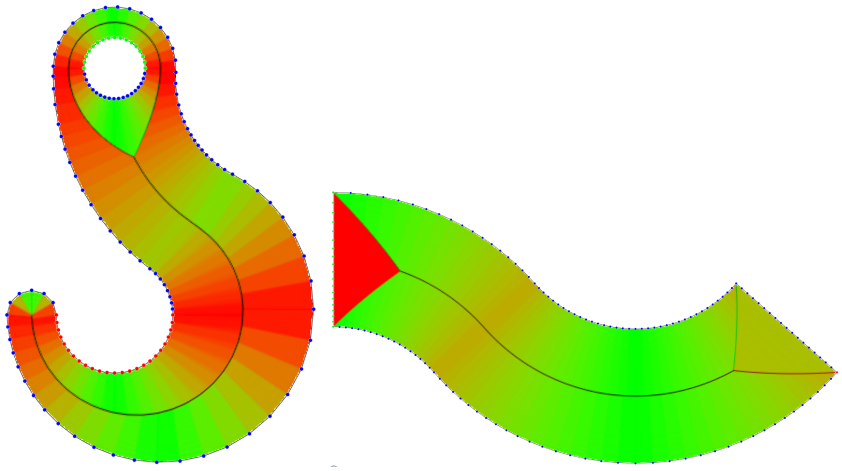
Algoritmo 2 Pseudocódigo para o *forward pass*

```
▷ V(x,y) retorna o vetor com a distância para a fronteira do vértice com coordenadas (x,y)
dir = {{−1,−1},{0,−1},{1,−1},{−1,0}}
for y = 1; y < f_y − 1; y++ do
    for x = 1; x < f_x − 1; x++ do
        p = P(x,y)                    ▷ Posição do vértice de coordenadas (x,y)
        i = argmin_j(||(P(x+dir_jx, y+dir_jy) − p) + V(x+dir_jx, y+dir_jy)||)
        V(x,y)= (P(x + dir_ix, y + dir_iy) − p) + V(x + dir_ix, y + dir_iy)
        F(x,y)= ||V(x,y)||
    end for
end for
```

Para ilustrar a implementação do campo de distância, a Figura 3.6 exibe alguns resultados do gradiente obtido a partir do campo de distância: a cor vermelha representa a componente X e a cor verde, a componente Y. A linha preta encontrada nos resultados é o eixo medial, região onde o gradiente não possui valor definido já que o ponto está equidistante a pelo menos dois pontos distintos na fronteira.



(a) Modelo Flor (b) Modelo Michell



(c) Modelo Gancho (d) Modelo Serpentine

Figura 3.6: Visualização do gradiente obtido através do campo de distância

3.3.2 Discretização do Domínio

Com o campo de distância, tem-se a direção que a propagação deve seguir a partir da fronteira, porém não se sabe o valor de espaçamento da propagação. Para esse cálculo é proposto um algoritmo que tenta respeitar a curvatura de seu contorno na propagação, levando em consideração a proporção que a propagação que os vizinhos terão.

Para isso, o primeiro valor de espaçamento do nó semente v é calculado da seguinte forma: computa-se a soma de todos os tamanhos das arestas as

quais v pertence, depois a soma de todos os tamanhos das arestas caso esses mesmos pontos estejam deslocados na direção das normais de seus elementos. Então é calculado o fator de proporção γ entre esses dois valores e definido o valor $\boldsymbol{j}_{0_v}$ como o primeiro valor de espaçamento de v, sendo ele proporcional a γ e ao tamanho da menor aresta ao qual v pertence. O Algoritmo 3 exibe o algoritmo proposto para o cálculo de $\boldsymbol{j}_{0_v}$ e γ, com a Figura 3.7 ilustrando um exemplo do cálculo de $\boldsymbol{j}_{0_v}$ para o caso 2D.

Os próximos valores de espaçamento são calculados com a seguinte equação:

$$\boldsymbol{j}_{i_v} = \boldsymbol{grad}_i * ||\boldsymbol{j}_{0_v}|| * \gamma^i, i > 0 \tag{3-4}$$

Algoritmo 3 Algoritmo usado para o cálculo de $\boldsymbol{j}_{0_v}$. No caso 2D, o elemento E é um segmento de reta e no 3D, E são triângulos

```
1: v é a semente atual
2: dist_0 = 0.0
3: dist_1 = 0.0
4: size = TamMenorSegReta(v)
5: grad_v = Gradiente(v)
6: for all Elemento E | v ∈ E do
7:     for all Nó p | p ∈ E do
8:         if p ≠ v then
9:             dist_0 = dist_0 + dist(v, p)
10:            dist_1 = dist_1 + dist(v + grad_v * size, p + Normal(E) * size)
11:        end if
12:    end for
13: end for
14: γ = (dist_1/dist_0)^2
15: j_{0_v} = grad_v * size * γ
```

A propagação do nó semente v é interrompida quando uma dessas condições acontece:

- $||\boldsymbol{j}_{i_v}|| \leq 0.2||\boldsymbol{j}_{0_v}||$. Isso ocorre porque a continuação da propagação para valores inferiores de $||\boldsymbol{j}_{i_v}||$ não agrega qualidade para o resultado da otimização.
- O novo nó criado está no eixo medial. Nesse caso, a propagação precisa parar pois o gradiente não está definido nessa região, impossibilitando a propagação.
- O gradiente do novo nó aponta no sentido contrário ao nó de origem. Isso representa que o nó passou pelo eixo medial, sendo o nó "levado" de volta ao eixo medial.

Como um dos fatores de parada da propagação é o nó ser criado na região do eixo medial, os nós sementes que se encontram nas quinas do contorno, isto

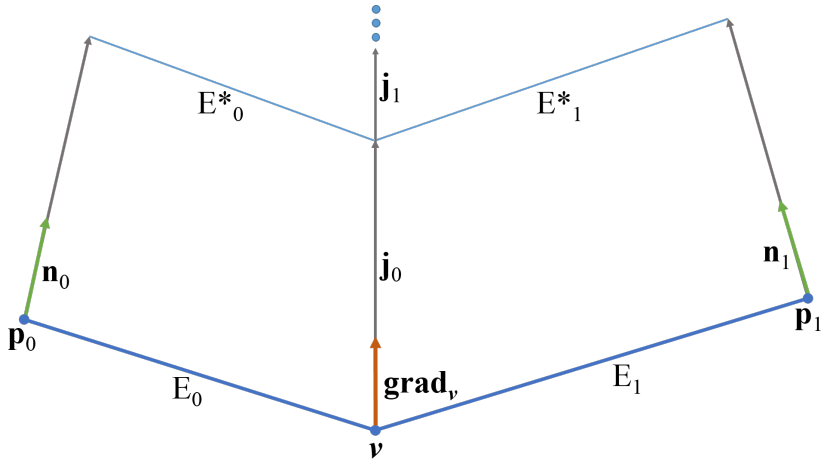


Figura 3.7: Exemplo 2D para o cálculo do vetor que será usado para os saltos da discretização. No exemplo, supondo $||E_0|| > ||E_1||$, temos em relação ao algoritmo presente no Algoritmo 3: $size = ||E_0||$, $dist_0 = ||E_0|| + ||E_1||$, $dist_1 = ||E_0^*|| + ||E_1^*||$, $\gamma = (||E_0^*|| + ||E_1^*||/||E_0|| + ||E_1||)^2$

é, a normal entre dois dos elementos ao qual o nó pertence possui um ângulo menor ou igual a 90°, não geram propagação.

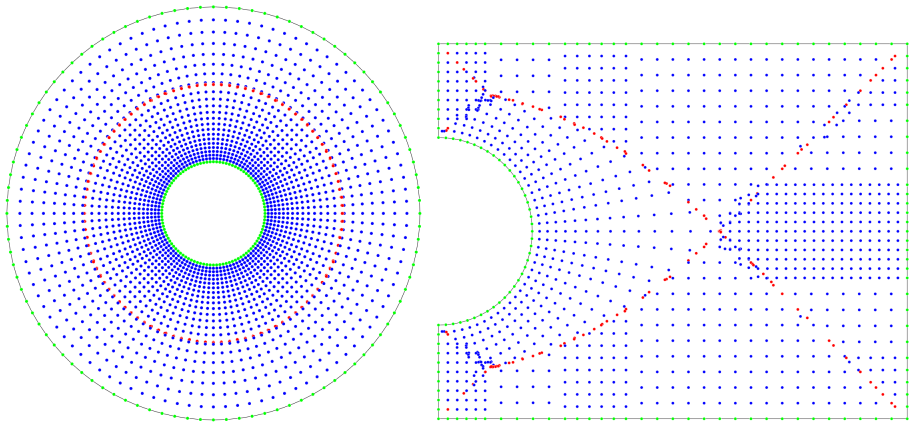
A Figura 3.8 exibe o resultado obtido, sendo os nós verdes as sementes, os nós azuis região com gradiente definido e os nós vermelhos região do eixo medial. Como pode-se observar, foi obtido um resultado que obedece ao contorno da fronteira, porém com um grande número de nós perto do eixo medial. Para diminuir a concentração de nós nessa região, foi implementada uma heurística para a remoção de nós.

3.3.3 Remoção de nós

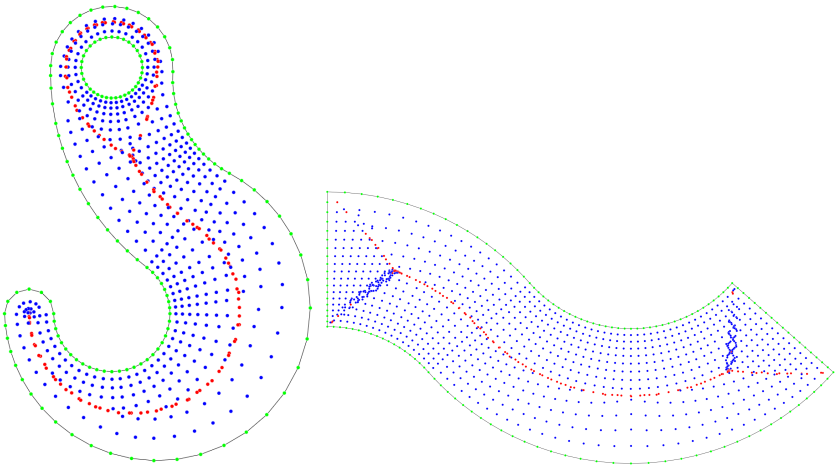
Para remover os nós, criou-se uma "zona de influência" para cada nó, isto é, uma região onde só um nó poderá existir. Além disso, também foram separados os casos dos nós que se encontram no eixo medial (os nós vermelhos da Figura 3.8) e os que não estão nessa região (os nós azuis da Figura 3.8).

Inicialmente, os nós azuis são ordenados pela ordem de sua criação. Isto é, o nó semente tem "Ordem 0", o nó criado a partir dele tem "Ordem 1", e assim sucessivamente. Depois, eles são percorridos verificando se há nós dentro de sua zona de influência, exemplificado na Figura 3.9. Supondo que se esteja processando o nó $\boldsymbol{v}$, são definidas as seguintes regras:

- O raio do círculo (ou esfera) de influência de $\boldsymbol{v}$ é proporcional ao valor de $||\boldsymbol{j}_{i_v}||$, sendo esse fator de proporcionalidade chamado de β
- Se dentro da zona de $\boldsymbol{v}$ existir algum nó $\boldsymbol{p}$ tal que $\boldsymbol{p}$ é da região do eixo medial, ou de Ordem superior a $\boldsymbol{v}$, o nó $\boldsymbol{p}$ será removido. Se $\boldsymbol{p}$ for da



(a) Modelo Flor (b) Modelo Michell



(c) Modelo Gancho (d) Modelo Serpentine

Figura 3.8: Resultado inicial da discretização do domínio. Os nós verdes são os nós sementes, os nós azuis são os da região onde o gradiente existe e os vermelhos são o da região do eixo medial.

mesma Ordem, porém se estiver a uma distância maior da fronteira do que $\boldsymbol{v}$, $\boldsymbol{p}$ será removido

Por fim, os nós do eixo medial que sobraram também são ordenados pela ordem de sua criação e é executada uma heurística semelhante à dos nós azuis, exemplificado na Figura 3.10, com as seguinte regras para o nó $\boldsymbol{v}$:

- Utiliza-se o mesmo valor β como raio de influência
- Se dentro da zona de $\boldsymbol{v}$ existir algum nó $\boldsymbol{p}$ tal que $\boldsymbol{p}$ é da região do eixo medial e de Ordem superior a $\boldsymbol{v}$, o nó $\boldsymbol{p}$ será removido. Se $\boldsymbol{p}$ for do eixo medial e da mesma Ordem, porém se encontrar a uma distância maior da fronteira do que $\boldsymbol{v}$, $\boldsymbol{p}$ será removido. Contudo, ao remover todos os nós $\boldsymbol{p}$'s, é calculada uma nova posição para $\boldsymbol{v}$, sendo o resultado da média ponderada das posições dos nós que ele removeu, com o peso sendo o inverso da Ordem.

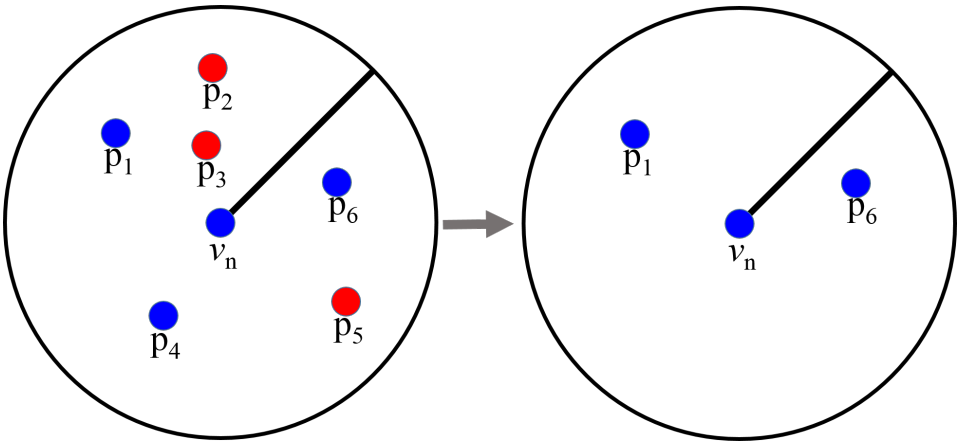


Figura 3.9: Exemplo de remoção dos nós azuis. Neste exemplo, a Ordem de p_1 e p_6 é menor do que v_n, por isso não são removidos.

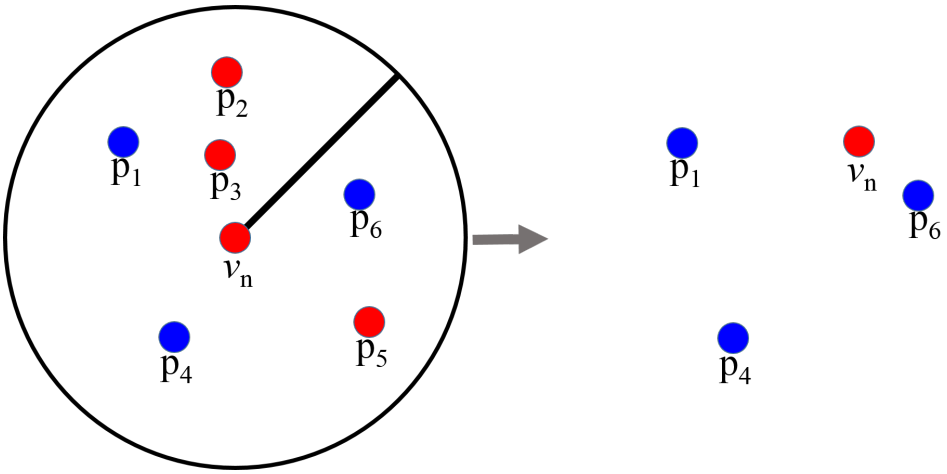


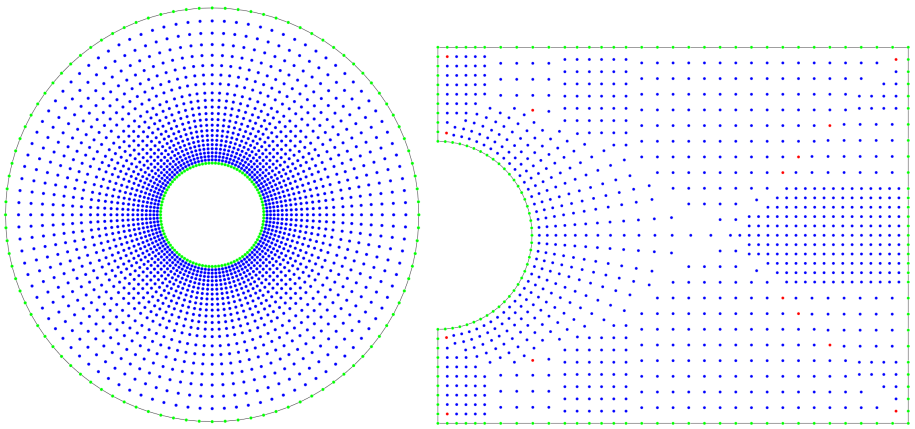
Figura 3.10: Exemplo de remoção dos nós vermelhos. O nó v_n não remove nenhum nó azul e, além disso, sua posição também é modificada levando em consideração a posição dos nós removidos.

A Figura 3.11 ilustra alguns resultados alcançados, para diferentes domínios. Nela, é possível ver que foi gerada uma boa discretização, seguindo o contorno definido pela fronteira. Além disso, os pontos também preenchem bem todo o interior do domínio.

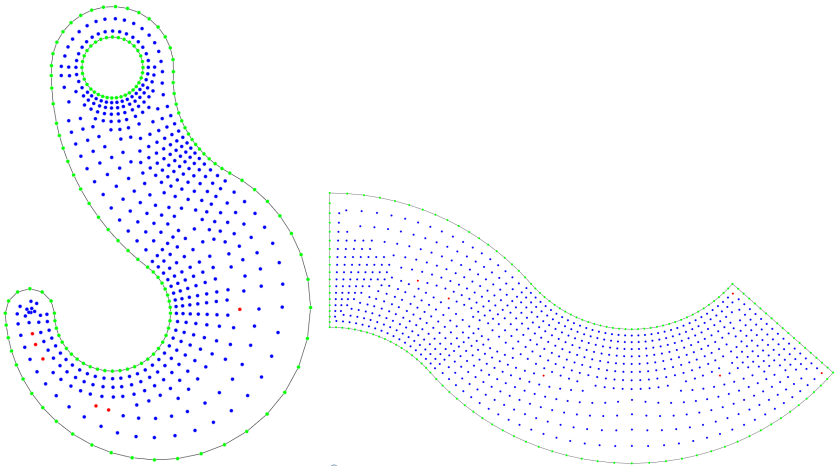
3.3.4 Estudo dos parâmetros

Durante o processo de geração dos nós, alguns parâmetros foram criados para sua realização. São eles:

- Fator de multiplicação τ para definir o tamanho da célula. Variando esse valor, a qualidade do campo de distância também é alterada.
- Fator de proporcionalidade β para a criação da área de influência e posterior remoção dos nós.



(a) Modelo Flor (b) Modelo Michell

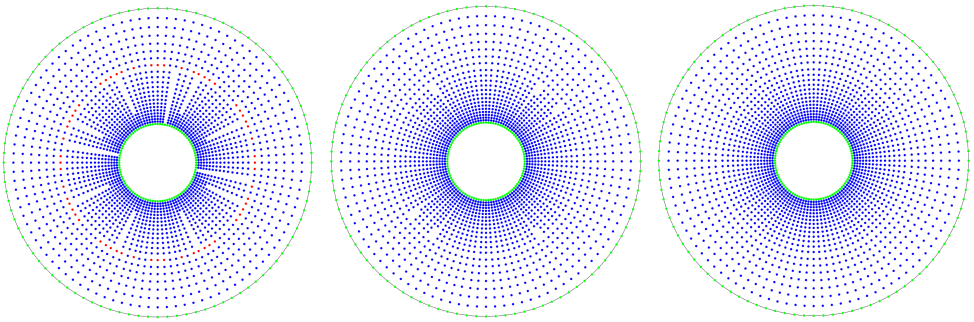


(c) Modelo Gancho (d) Modelo Serpentine

Figura 3.11: Resultado final da discretização do domínio. Os nós verdes são os nós sementes, os nós azuis são os da região onde o gradiente existe e os vermelhos são o da região do eixo medial.

A qualidade da discretização do domínio está ligada diretamente ao tamanho das células de sua grade. Quanto mais refinada a grade, melhor sua qualidade. A Figura 3.12 exibe os resultados obtidos para a escolha de τ com valores 1.0, 0.5 e 0.25. Essa variação ocorre devido à precisão no cálculo do gradiente, que é mais preciso quanto mais discretizada for a malha. Testes realizados com diferentes os modelos mostraram que um valor de τ menor que 0.25 não altera o resultado final da geração dos nós, somente aumentando o custo computacional da geração.

Apesar de ser difícil definir um valor único para todos os modelos, testes realizados mostraram que o valor $\beta = 0.8$ foi satisfatório para todos os exemplos testados.



(a) $\tau = 1.0$ (b) $\tau = 0.5$ (c) $\tau = 0.25$

Figura 3.12: Resultado da variação do tamanho da célula na geração dos nós

3.4 Geração das barras

Com os nós gerados, é preciso conectá-los através de barras para gerar a malha densa de barras. Para isso, foi implementada uma heurística semelhante à da remoção dos nós. Porém, ao invés de remover os nós dentro da zona de influência, criam-se barras entre esses nós.

Em contraste ao GRAND, que utiliza um conceito topológico para a definição da densidade da malha, este trabalho propõe um conceito geométrico para a escolha da densidade. Para isso, defini-se um raio r para cada nó v da malha com a forma:

$$r = ||\boldsymbol{j}_{i_v}|| * \sigma \tag{3-5}$$

em que r é proporcional ao espaçamento de cada nó e a um *raio de conectividade* σ.

As barras candidatas associadas ao nó v são definidas como todas as barras originadas em v contidas dentro do círculo definido pelo raio r da Equação 3-5. O algoritmo proposto para a geração das barras, inicialmente, adiciona todas as barras candidatas de todos nós em um conjunto. Depois, as barras são ordenadas pelo seu tamanho, para que as menores tenham prioridades no processo de adição na malha. Por fim, esse conjunto é percorrido tentando adicionar a barra a malha. Dado que uma barra é adicionada, ela nunca será removida. Os testes para saber se uma barra é ou não adicionada são:

- Se a barra é de um nó de dentro domínio, verifica-se se a mesma não ultrapassa a fronteira do domínio. Para testar isso, foi usado novamente o caminhamento na grade (15) para verificar se a barra não colide com os elementos (segmento de reta/triângulos) definidos na fronteira. Colidindo, a barra é descartada
- Se a barra é de um nó da fronteira, é necessário que a barra esteja na

mesma direção da normal do domínio. Se não estiver, é porque a barra está sendo criada para fora do domínio, sendo então descartada

– Se a barra for colinear a outras barras, ela é então descartada. Para isso, assim como no GRAND, é definido um ângulo $ColTol$ máximo de tolerância, em que na Figura 3.13, a barra tracejada só será descartada se $cos(\beta_2) < ColTol$ e $cos(\beta_1) < ColTol$

Passando por todos esses testes sem ser descartada, a barra é então adicionada a malha. O Algoritmo 4 exibe o pseudocódigo do algoritmo proposto para a geração das barras.

Algoritmo 4 Algoritmo proposto para a geração de barras

```
1: n é o nível de conectividade
2: Conjunto barras_candidatas;
3: for all Nós v da Malha do
4:     barras_candidatas.Adiciona(BarrasCandidatas(v, n))
5: end for
6: OrdenaPorTamanho(barras_candidatas)
7: for all Barras b em barras_candidatas do
8:     TentaAdicionarBarraNaMalha(b)
9: end for
```

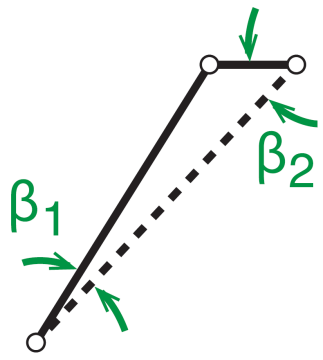
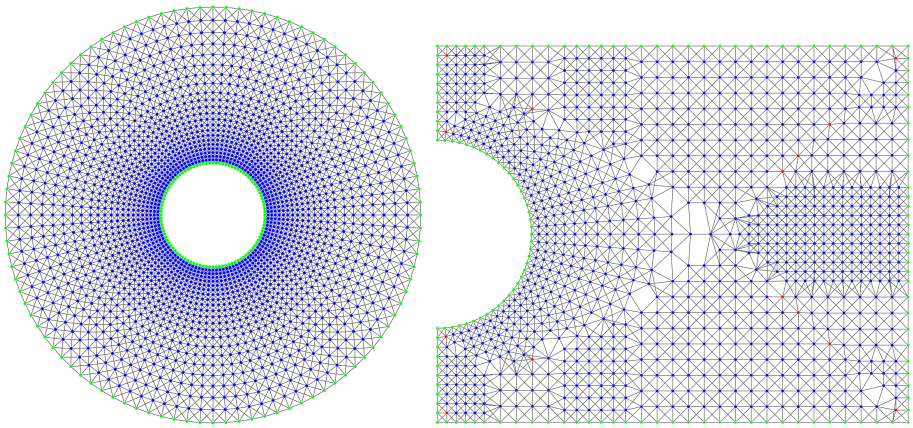
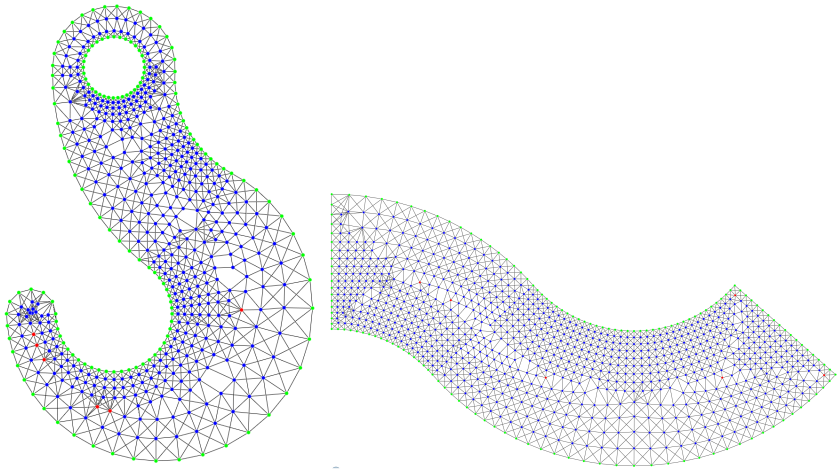


Figura 3.13: Teste de colinearidade. Se $cos(\beta_2) < ColTol$ e $cos(\beta_1) < ColTol$, então a barra pontilhada não será adicionada.

A Figura 3.14 ilustra resultados de geração das barras para o raio de conectividade igual a 1.6. Apesar de não fazer sentido um nível tão baixo para a otimização, ele é mostrado como um resultado deste trabalho para ser possível visualizar a malha densa de barras gerada, já que raios mais elevados não são possíveis visualizar. A Figura 3.15 mostra o resultado da variação desse raio nos valores 1.1, 1.6 e 2.1, resultando em malhas cada vez mais densas para valores maiores.

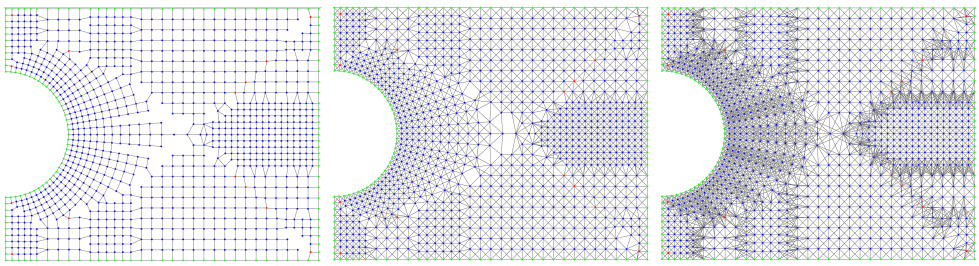


(a) Modelo Flor (b) Modelo Michell



(c) Modelo Gancho (d) Modelo Serpentine

Figura 3.14: Resultado da geração de barras para o grau de conectividade 1



(a) $\sigma = 1.1$ (b) $\sigma = 1.6$ (c) $\sigma = 2.1$

Figura 3.15: Resultado da variação do fator σ

4
Otimização Topológica Estrutural de Barras

Neste capítulo, é apresentado o desenvolvimento da otimização topológica estrutural de barras deste trabalho. A Seção 4.1 explica quais são as premissas necessárias e como a otimização topológica pode ser vista como um problema de programação linear, com a Seção 4.2 exemplificando alguns métodos para se resolver problemas de programação linear e explicando qual o método implementado para esta dissertação. Por fim, na Seção 4.3, é apresentada como foi realizada a implementação da otimização topológica no *framework* TopSim (4).

4.1
Formulação da Otimização Topológica Estrutural de Barras

Esta seção apresenta resumidamente a formulação do método da malha densa de barras (*ground strucuture method*, GSM) baseada na análise plástica, a qual será utilizada neste trabalho. Um maior aprofundamento da análise plástica e elástica do GSM é apresentado em (5).

O objetivo da otimização utilizada nesta dissertação é minimizar o volume da treliça, também satisfazendo as equações de equilíbrio e as restrições impostas. A otimização somente modifica a área de secção transversal de cada barra, minimizando assim o volume final da malha, sem alterar as posições dos nós. Com isso, formula-se (6, 9, 10):

$$\begin{aligned} \min_{\boldsymbol{a}} \quad & V = \boldsymbol{l}^T \boldsymbol{a} \\ \text{s.t.} \quad & \boldsymbol{B}^T \boldsymbol{n} = \boldsymbol{f} \\ & -\sigma_C a_i \leq n_i \leq \sigma_T a_i, \; i = 1, 2 \ldots N_b \end{aligned} \tag{4-1}$$

Onde:

- N_b é o número de barras na malha
- V é o volume da treliça
- σ_C e σ_T são os limites da tensão em compressão e tração
- a_i, l_i, σ_i, f_i e n_i são a área da secção transversal, o comprimento, a tensão, a força externa aplicada e a força interna (axial) da i-ésima barra

– $\boldsymbol{B}^T$ é a matriz de equilíbrio nodal, construída a partir dos cosenos direcionais de cada membro

$\boldsymbol{B}^T$ é uma matriz de tamanho $N_{dof} \times N_b$, em que:

– N_{dof} é o número total de graus de liberdade da malha, sendo $N_{dof} = 2N_n - N_{sup}$ para o caso 2D e $N_{dof} = 3N_n - N_{sup}$ para o 3D

– N_n é o número de nós da malha densa de barras

– N_{sup} é o número de graus de liberdade fixo

Essa formulação, baseada na análise plástica, somente garante o equilíbrio das forças (6, 9, 10). Adicionando as variáveis de folga (*slack variables*) $\boldsymbol{s}^+$ e $\boldsymbol{s}^-$ nas restrições de tensão (6, 9, 10), as desigualdades são transformadas em igualdades (5):

$$\begin{aligned} a_i &= \frac{s_i^+}{\sigma_T} + \frac{s_i^-}{\sigma_C} \\ n_i &= s_i^+ - s_i^- \end{aligned} \tag{4-2}$$

Transformando assim o problema da Equação 4-1 em um problema de programação linear da forma:

$$\begin{aligned} \min_{s_i^+, s_i^+} \quad & V = \boldsymbol{l}^T(\frac{\boldsymbol{s}^+}{\sigma_T} + \frac{\boldsymbol{s}^-}{\sigma_C}) \\ \text{s.t.} \quad & \boldsymbol{B}^T(\boldsymbol{s}^+ - \boldsymbol{s}^-) = \boldsymbol{f} \\ & s_i^+, s_i^- \geq 0 \end{aligned} \tag{4-3}$$

Se a razão do limite de tensão for definido como $k = \frac{\sigma_T}{\sigma_C}$, então a Equação 4-3 fica:

$$\begin{aligned} \min_{s_i^+, s_i^+} \quad & V^* = \frac{V}{\sigma_T} = \boldsymbol{l}^T(\boldsymbol{s}^+ + k\boldsymbol{s}^-) \\ \text{s.t.} \quad & \boldsymbol{B}^T(\boldsymbol{s}^+ - \boldsymbol{s}^-) = \boldsymbol{f} \\ & s_i^+, s_i^- \geq 0 \end{aligned} \tag{4-4}$$

Escrevendo a Equação 4-4 na notação matricial, tem-se:

$$\begin{aligned} \min_{s_i^+, s_i^+} \quad & V^* = \frac{V}{\sigma_T} = \underset{1\times 2N_b}{\begin{bmatrix} \boldsymbol{l}^T & k\boldsymbol{l}^T \end{bmatrix}} \underset{2N_b\times 1}{\begin{bmatrix} \boldsymbol{s}^+ \\ \boldsymbol{s}^- \end{bmatrix}} \\ \text{s.t.} \quad & \underset{N_{dof}\times 2N_b}{\begin{bmatrix} \boldsymbol{B}^T & -\boldsymbol{B}^T \end{bmatrix}} \underset{2N_b\times 1}{\begin{bmatrix} \boldsymbol{s}^+ \\ \boldsymbol{s}^- \end{bmatrix}} = \underset{N_{dof}\times 1}{\boldsymbol{f}} \\ & s_i^+, s_i^- \geq 0 \end{aligned} \tag{4-5}$$

Sendo a Equação 4-5 a forma final (5, 11, 12, 13) utilizada neste trabalho. É importante notar que a adição das variáveis $\boldsymbol{s}^+$ e $\boldsymbol{s}^-$ dobram o número de incógnitas do problema. Isto é, enquanto que na Equação 4-1 as incógnitas do problema são as áreas da secção transversal de cada barra presente na malha, o

número de incógnitas para o problema de programação linear que será resolvido neste trabalho é o dobro do número de barras da malha.

4.2 Programação Linear

Programação linear (PL) é uma área da matemática que busca métodos para resolver problemas de otimização (minimização/maximização), possuindo uma função objetivo linear, sujeita a equações e inequações lineares como restrições.

Um dos métodos mais conhecidos (22) para resolver problemas de programação linear é o método do Simplex, desenvolvido por Dantzig (23). O espaço da solução de um problema PL é um poliedro convexo; com isso, o método do Simplex propõe o caminhamento pelos vértices desse poliedro. A partir de um vértice, busca-se um vértice vizinho que melhore a função objetivo, parando a execução quando não existir mais um vértice vizinho que melhore a solução. Esse método é utilizado em várias problemas PL, porém não é indicado para problemas de larga escala (11), pois sua complexidade é exponencial.

Khachiyan mostrou que o problema PL poderia ser resolvido em tempo polinomial (24), porém, em termos práticos, continuou sendo melhor utilizar o método Simplex do que o proposto por Khachiyan. No entanto, seu trabalho impulsionou novas pesquisas na resolução de problemas LP e Karmarkar propôs um novo método, chamado de Método dos Pontos Interiores (25) (*interior points methods*, IPM), sendo posteriormente aperfeiçoado por Mehrotra (26), Wright (27) e Nocedal e Wright (28).

Neste caso, o IPM mostrou-se ser superior ao Simplex também em termos práticos. Enquanto o Simplex caminha pelos vértices, o IPM encontra a resolução do problema atravessando o interior do espaço da solução. Os métodos mais populares para o IPM são: primal-afim-escala e primal-dual (11), com a versão primal-dual convergindo mais rápido que a primal-afim-escala, porém ocupando mais memória.

Neste trabalho, foi implementado o método de pontos interiores primal-dual com o algoritmo de preditor-corretor proposto por Mehrotra (26).

4.2.1 Método de Pontos Interiores Primal-Dual com algoritmo Preditor-Corretor

O problema de programação linear em sua forma padrão é:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq 0 \end{aligned} \tag{4-6}$$

Em que $\boldsymbol{c}, \boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$ e $\boldsymbol{A}$ é uma matriz $m \times n$. O problema dual da Equação 4-6 é:

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & \boldsymbol{b}^T\boldsymbol{\lambda} \\ \text{s.t.} \quad & \boldsymbol{A}^T\boldsymbol{\lambda} + \boldsymbol{s} = \boldsymbol{c} \\ & \boldsymbol{s} \geq 0 \end{aligned} \tag{4-7}$$

Em que $\boldsymbol{\lambda} \in \mathbb{R}^m$ e $\boldsymbol{s} \in \mathbb{R}^n$. Supondo $\boldsymbol{x}^*$ como a solução do problema 4-6, então existem $\boldsymbol{s}^*$ e $\boldsymbol{\lambda}^*$ que satisfazem as condições de Karush–Kuhn–Tucker (KKT):

$$\begin{aligned} \boldsymbol{A}^T\boldsymbol{\lambda}^* + \boldsymbol{s}^* &= \boldsymbol{c} \\ \boldsymbol{A}\boldsymbol{x}^* &= \boldsymbol{b} \\ x_i^* s_i^* &= 0, i = 1, \ldots, n \\ (\boldsymbol{x}^*, \boldsymbol{s}^*) &\geq 0 \end{aligned} \tag{4-8}$$

Sendo esta a condição necessária para determinar que ($\boldsymbol{x}^*$, $\boldsymbol{s}^*$, $\boldsymbol{\lambda}^*$) é a solução ótima do primal-dual. Para encontrar a resolução do problema, utiliza-se o método de Newton para as condições KKT, de forma que:

$$\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{s}) = \begin{bmatrix} \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \\ \boldsymbol{A}^T\boldsymbol{\lambda} + \boldsymbol{s} - \boldsymbol{c} \\ \boldsymbol{X}\boldsymbol{S}\boldsymbol{e} \end{bmatrix} = 0 \tag{4-9}$$

$$\boldsymbol{J}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{s}) \begin{bmatrix} \Delta\boldsymbol{\lambda} \\ \Delta\boldsymbol{x} \\ \Delta\boldsymbol{s} \end{bmatrix} = -\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{s}) \tag{4-10}$$

sendo $\boldsymbol{X} = \text{diag}(x_1, \ldots, x_n)$, $\boldsymbol{S} = \text{diag}(s_1, \ldots, s_n)$, $\boldsymbol{e} = [1, \ldots, 1]^T \in \mathbb{R}^n$, $\boldsymbol{J}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{s})$ a matriz Jacobiana de $\boldsymbol{F}$ e $(\Delta\boldsymbol{\lambda}, \Delta\boldsymbol{x}, \Delta\boldsymbol{s})$ as direções de Newton.

O método primal-dual possui a abordagem conhecida como função barreira logarítmica, em que, a cada iteração t, são gerados os vetores ($\boldsymbol{x}^t$, $\boldsymbol{s}^t$, $\boldsymbol{\lambda}^t$), com $x_i^t s_i^t = \sigma\mu$, $i = 1, \ldots, n$, de forma que, durante a execução, $\sigma\mu \to 0$ e ($\boldsymbol{x}^t$, $\boldsymbol{s}^t$, $\boldsymbol{\lambda}^t$) $\to$ ($\boldsymbol{x}^*$, $\boldsymbol{s}^*$, $\boldsymbol{\lambda}^*$). Com isso, a Equação 4-10 fica:

$$\begin{bmatrix} 0 & \boldsymbol{A} & 0 \\ \boldsymbol{A}^T & 0 & \boldsymbol{I} \\ 0 & \boldsymbol{S} & \boldsymbol{X} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\lambda} \\ \Delta\boldsymbol{x} \\ \Delta\boldsymbol{s} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \\ -\boldsymbol{A}^T\boldsymbol{\lambda} - \boldsymbol{s} + \boldsymbol{c} \\ -\boldsymbol{X}\boldsymbol{S}\boldsymbol{e} + \sigma\mu\boldsymbol{e} \end{bmatrix} \tag{4-11}$$

No algoritmo Preditor-Corretor proposto por Mehrotra, inicialmente é calculada uma direção preditor, sendo em seguida computado, o termo σ para melhorar o caminho seguido pelo algoritmo e, por fim, é calculada uma direção corretor que tenta compensar a não-linearidade do preditor. Tanto para o

cálculo do corretor quanto para o preditor, é necessário resolver um sistema linear da forma $\boldsymbol{Ax} = \boldsymbol{b}$. Porém, os dois casos possuem a mesma matriz de coeficientes $\boldsymbol{A}$, sendo assim a fatoração é realizada apenas uma vez pelo solver.

O Algoritmo 5 exibe cada passo do algoritmo Preditor-Corretor (30). O valor ϵ para a parada da execução para este trabalho é $\epsilon = 10^{-8}$ (29). A Linha 11 é a computação do sistema linear para o cálculo da direção do preditor, a Linha 18 é a heurística proposta por Mehrotra e a Linha 22 é a resolução do sistema $\boldsymbol{Ax} = \boldsymbol{b}$ para a direção do corretor, sendo a direção final a soma do preditor com o corretor.

Algoritmo 5 Dual-Primal Corretor-Preditor

1: Dado $(\boldsymbol{x}^0, \boldsymbol{\lambda}^0, \boldsymbol{s}^0)$
2: t=0
3: $\boldsymbol{e} = [\boldsymbol{1, \ldots, 1}]^{\boldsymbol{T}}$
4: **while** $\frac{|\boldsymbol{c}^T\boldsymbol{x}^t - \boldsymbol{b}^T\boldsymbol{\lambda}^t|}{1+|\boldsymbol{c}^T\boldsymbol{x}^t|} \leq \epsilon$ **do**
5: $\quad \boldsymbol{r}_b = \boldsymbol{Ax}^t - \boldsymbol{b}$
6: $\quad \boldsymbol{r}_c = \boldsymbol{A}^T\boldsymbol{\lambda}^t + \boldsymbol{s}^t - \boldsymbol{c}$
7: $\quad \boldsymbol{X} = \text{diag}(x_1^t, \ldots, x_n^t)$
8: $\quad \boldsymbol{S} = \text{diag}(s_1^t, \ldots, s_n^t)$
9: $\quad \boldsymbol{S}^{-1} = \text{diag}(1/s_1^t, \ldots, 1/s_n^t)$
10: $\quad \boldsymbol{D}^2 = \text{diag}(x_1^t/s_1^t, \ldots, x_n^t/s_n^t)$
11: $\quad$ Solve: $\boldsymbol{AD}^2\boldsymbol{A}^T\Delta\boldsymbol{\lambda}^{aff} = -\boldsymbol{r}_b - \boldsymbol{A}(\boldsymbol{S}^{-1}\boldsymbol{X}\boldsymbol{r}_c - \boldsymbol{S}^{-1}\boldsymbol{XSe})$
12: $\quad \Delta\boldsymbol{s}^{aff} = -\boldsymbol{r}_c - \boldsymbol{A}^T\boldsymbol{\lambda}^{aff}$
13: $\quad \Delta\boldsymbol{x}^{aff} = -\boldsymbol{S}^{-1}(\boldsymbol{XSe} + \boldsymbol{X}\Delta\boldsymbol{s}^{aff})$
14: $\quad \alpha_{aff}^{pri} = Min[1, \{x_i^t/\Delta x_i^{aff}, \Delta x_i^{aff} < 0\}]$
15: $\quad \alpha_{aff}^{dual} = Min[1, \{s_i^t/\Delta s_i^{aff}, \Delta s_i^{aff} < 0\}]$
16: $\quad \mu = \frac{\sum x_i^t s_i^t}{n}$
17: $\quad \mu_{aff} = \frac{(\boldsymbol{x}^t + \alpha_{aff}^{pri}\Delta\boldsymbol{x}^{aff})^T(\boldsymbol{s}^t + \alpha_{aff}^{dual}\Delta\boldsymbol{s}^{aff})}{n}$
18: $\quad \sigma = (\frac{\mu_{aff}}{\mu})^3$
19: $\quad \Delta\boldsymbol{X}^{aff} = \text{diag}(\Delta x_1^{aff}, \ldots, \Delta x_n^{aff})$
20: $\quad \Delta\boldsymbol{S}^{aff} = \text{diag}(\Delta s_1^{aff}, \ldots, \Delta s_n^{aff})$
21: $\quad \boldsymbol{r}_{xs} = -\sigma\mu\boldsymbol{e} + \Delta\boldsymbol{X}^{aff}\Delta\boldsymbol{S}^{aff}$
22: $\quad$ Solve: $\boldsymbol{AD}^2\boldsymbol{A}^T\Delta\boldsymbol{\lambda}^{cor} = \boldsymbol{AS}^{-1}\boldsymbol{r}_{xs}$
23: $\quad \Delta\boldsymbol{s}^{cor} = -\boldsymbol{A}^T\boldsymbol{\lambda}^{cor}$
24: $\quad \Delta\boldsymbol{x}^{cor} = -\boldsymbol{S}^{-1}(\boldsymbol{r}_{xs} + \boldsymbol{X}\Delta\boldsymbol{s}^{cor})$
25: $\quad (\Delta\boldsymbol{x}^t, \Delta\boldsymbol{\lambda}^t, \Delta\boldsymbol{s}^t) = (\Delta\boldsymbol{x}^{aff}, \Delta\boldsymbol{\lambda}^{aff}, \Delta\boldsymbol{s}^{aff}) + (\Delta\boldsymbol{x}^{cor}, \Delta\boldsymbol{\lambda}^{cor}, \Delta\boldsymbol{s}^{cor})$
26: $\quad \alpha_t^{pri} = Min[1, \{x_i^t/\Delta x_i^t, \Delta x_i^t < 0\}]$
27: $\quad \alpha_t^{dual} = Min[1, \{s_i^t/\Delta s_i^t, \Delta s_i^t < 0\}]$
28: $\quad \boldsymbol{x}^{t+1} = \boldsymbol{x}^t + \alpha_t^{pri}\Delta\boldsymbol{x}^t$
29: $\quad (\boldsymbol{\lambda}^{t+1}, \boldsymbol{s}^{t+1}) = (\boldsymbol{\lambda}^t, \boldsymbol{s}^t) + \alpha_t^{dual}(\Delta\boldsymbol{\lambda}^t, \Delta\boldsymbol{s}^t)$
30: $\quad$ t = t+1
31: **end while**
32: **return** $\boldsymbol{x}^t$

4.3 Implementação no framework TopSim

O *framework* TopSim é uma ferramenta desenvolvida para resolver problemas de análise numérica em larga escala (4). Sua arquitetura é baseada em *plugins*, isto é, cada componente necessário para a execução da análise numérica pode ser desenvolvido de forma isolada e com algoritmos especializados, garantindo assim um ambiente flexível e fácil de expandir. O núcleo dessa ferramenta é mínimo, contendo apenas um módulo gerenciador de *plugin*, responsável por carregá-los em tempo de execução do programa, usando para isso somente um arquivo de configuração de entrada.

Todas as funcionalidades necessárias para uma determinada aplicação são definidas dentro dos *plugins*, sem a necessidade de mudar seu núcleo. Os *plugins* disponibilizam sua interface para a comunicação entre os componentes, podendo um *plugin* exigir a existência de outro para sua execução. Pode-se assim gerar sistemas maiores e mais complexos, porém sem ocorrer a perda de desempenho para isso, já que cada *plugin* é especializado para sua tarefa. A única exceção na arquitetura desenvolvida pelo TopsSim em relação ao uso de *plugins* é na representação do modelo. Neste caso, não é possível desenvolver um *plugin*, sendo utilizada a biblioteca TopS (19) para este fim.

TopS é uma estrutura de dados topológica compacta desenvolvida para a representação de malha de elementos finitos, em que elementos e nós são as únicas entidades topológicas explicitamente salvas na memória. Além disso, essa biblioteca é considerada completa no sentido de que todas as adjacências topológicas podem ser recuperadas num tempo proporcional às entidades recuperadas.

A Figura 4.1 ilustra como o *framework* TopSim é usado para resolver o problema de otimização topológica estrutural de barras neste trabalho, com os componentes em cinza sendo os *plugins* desenvolvidos neste trabalho, os *plugins* em azul já foram previamente implementados na TopSim e os componentes em verdes sendo desenvolvidos por terceiros. Inicialmente, o *host* chama o *plugin* de análise *ground structure method* (GSM). Além de realizar a análise, ele também gerencia a leitura e a escrita dos dados, usando para isso outros dois *plugins*. O *plugin* de leitura carrega o arquivo contendo a malha densa de barras, transferindo-a para a biblioteca TopS, criando nela os nós, barras e restrições. Já o *plugin* de saída escreve o resultado da análise num arquivo de formato *neutral file* (37).

Para realizar a otimização, o GSM monta e resolve um problema de programação linear. Na montagem, é utilizado o *plugin* de numeração DOF, que numera cada grau de liberdade livre. Para resolver o problema de programa-

ção linear, é utilizado o *plugin* de otimização IPM, que implementa o método dos pontos interiores (*interior point method*). O IPM, devido ao tamanho dos problemas resolvidos neste trabalho, utiliza um *plugin* de matriz esparsa e um *plugin* de *solver* $\boldsymbol{Ax} = \boldsymbol{b}$.

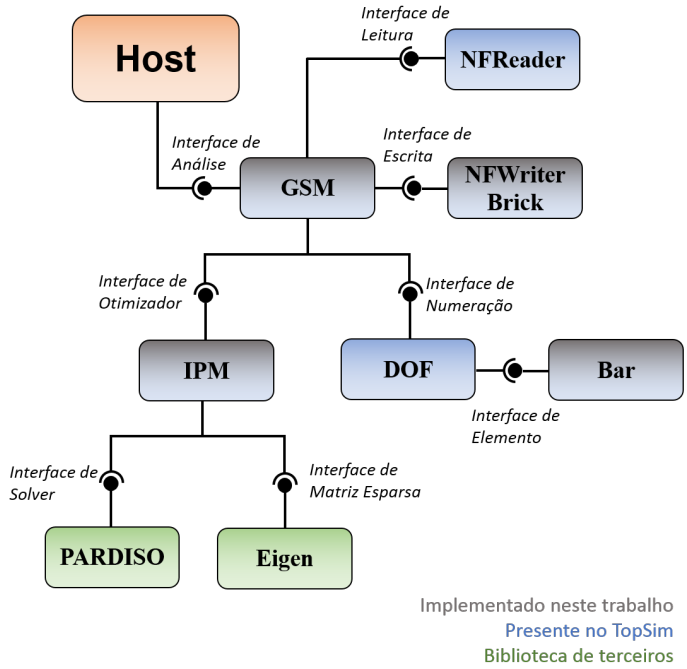


Figura 4.1: Especialização do TopSim para a otimização topológica estrutural de barras

O *plugin* GSM implementa a formulação descrita na Seção 4.1 e o IPM, o método explicado na Seção 4.2. Foi utilizada a biblioteca PARDISO (21) como o *solver* usado pelo IPM, sendo a biblioteca Eigen (20) utilizada para a matriz esparsa.

Para a visualização do resultado, foi utilizado a ferramenta de pós-processamento de modelos de elementos finitos chamada Pos3D (36). Para isso, o *plugin* NFWriterBrick escreve no arquivo de saída as barras como um elemento de formato Brick20, ilustrado na Figura 4.2.

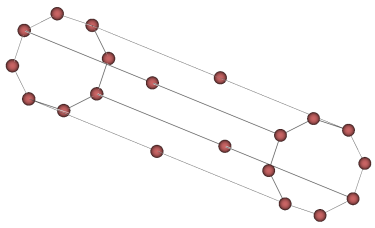


Figura 4.2: Barra como um elemento Brick20

5
Resultados

Neste capítulo, são apresentados os resultados obtidos com a geração da malha densa de barras proposta e da implementação da otimização topológica estrutural de barras no TopSim. Na Seção 5.1 é feita a verificação da qualidade da malha densa e da otimização, tanto visualmente quanto numericamente. A Seção 5.2 apresenta o desempenho obtido para a geração e para otimização de modelos com dezenas de milhões de barras. Por fim, a Seção 5.3 exibe resultados para outros modelos testados.

5.1
Verificação

Para verificar a qualidade da malha densa de barras gerada e a corretude da otimização implementada por este trabalho, foram feitas comparações com resultados obtidos utilizando o software GRAND da seguinte forma:

- Malha densa de barras e otimização feitas no GRAND
- Malha densa de barras gerada através do método proposto, com a otimização sendo realizada no GRAND
- Malha densa de barras e otimização feitas usando a implementação desta dissertação

Para se tentar uma comparação justa, foram geradas malhas com um número próximo de nós e barras para o mesmo domínio.

Como no GRAND, para a visualização do resultado final da otimização, é definido um valor de *cutoff* para a exibição das barras. Com isso, somente as barras com área de seção transversal $\frac{a_i}{max(a)} > cutoff$ são exibidas, sendo utilizado o valor de $cutoff = 0.002$ para todos os modelos 2D e $cutoff = 0.005$ para todos os casos 3D. O renderizador utilizado para o caso em que é utilizada a otimização do GRAND é o visualizador presente no Matlab, enquanto a otimização implementenda nesta dissertação utiliza o Pos3D (36) para visualizar a estrutura final.

Como explicado no Capítulo 3, é necessário definir os valores de dois parâmetros para a geração da malha densa de barras proposta, que são:

- τ: Fator que define o tamanho da célula da grade regular

– β: Fator do raio de influência do nó no processo de remoção dos nós vizinhos

Todos os resultados foram obtidos com os parâmetros possuindo os valores: $\tau = 0.2$ e $\beta = 0.8$. A qualidade visual e o volume final das malhas obtidas após as otimizações são similares, o que confirma que a malha densa de barras proposta nesta dissertação apresenta qualidade semelhante à do GRAND, porém sem a necessidade de uma malha de elementos finitos e zonas de restrições em sua entrada, além de certificar a implementação da otimização topológica estrutural de barras no TopSim.

5.1.1 Modelo Flor

O modelo Flor foi analisado por Zegard e Paulino (5), com o domínio sendo apresentado na Figura 5.1(a). A Figura 5.1(b) apresenta o resultado com a geração de malha e otimização sendo executadas pelo GRAND e as Figuras 5.1(c) e 5.1(d) mostram os resultados com a geração de malha densa desta dissertação, com o otimizador do GRAND e com a implementação no TopSim, respectivamente. A Tabela 5.1 exibe a quantidade de nós e barras de cada modelo utilizado. A Tabela 5.2 mostra o volume final obtido para cada caso executado. Para o GRAND, foi utilizado nível de conectividade com valor 8 e, para este trabalho, raio de conectividade igual a 7.4.

Um possível motivo para o volume obtido pela malha densa de barras gerada neste trabalho ser ligeiramente menor que a malha criada pelo GRAND é a diferença no número de nós e barras, já que a malha desta dissertação é mais discretizada e possui um número maior de barras. Já a visualização da estrutura final apresenta padrões diferentes de "pétalas" para cada malha densa de barras inicial.

	Nós	Barras
GRAND	2100	221800
Método Proposto	2300	222944

Tabela 5.1: Número de nós e barras da malha densa de barras para o Modelo Flor utilizados para comparação do resultado da otimização entre o GRAND e o proposto neste trabalho

Geração Malha Densa/Otimização	Volume
GRAND/GRAND	13.9674
Método Proposto/GRAND	13.8883
Método Proposto/Método Proposto	13.8883

Tabela 5.2: Volume obtido na otimização para cada malha densa de barras e o otimizador utilizado

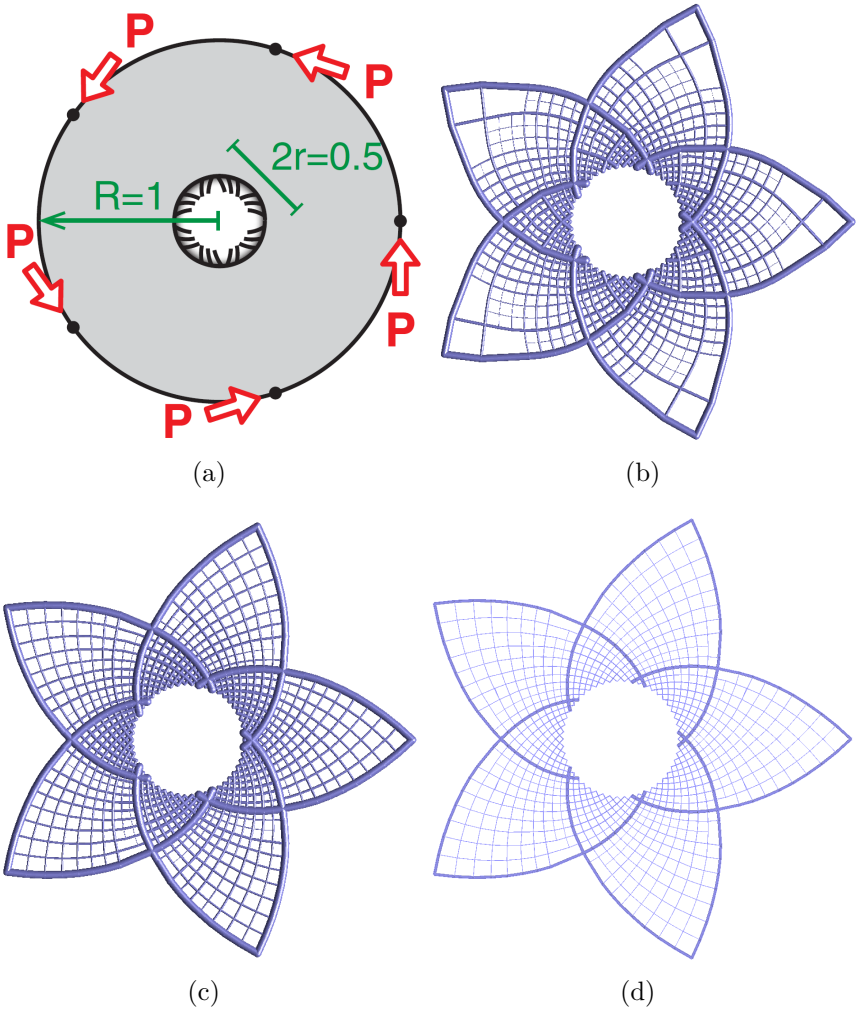


Figura 5.1: Resultado da otimização topológica do Modelo Flor. (a) Geometria, forças aplicadas e condições de contorno. (b) Resultado obtido com a malha densa de barras gerada pelo GRAND, também utilizando o GRAND como otimizador. (c) Resultado obtido com a malha densa de barras gerada pela proposta deste trabalho, utilizando o GRAND como otimizador. (d) Resultado obtido com a malha densa de barras gerada pela proposta deste trabalho, utilizando o otimizador desta dissertação.

5.1.2 Modelo Michell

O modelo Michell possui sua solução analítica desenvolvida por Michell (1), com o domínio sendo apresentado na Figura 5.2(a). A Figura 5.2(b)

apresenta o resultado com a geração de malha e otimização sendo executadas pelo GRAND e as Figuras 5.2(c) e 5.2(d) mostram os resultados com a geração de malha densa desta dissertação, com o otimizador do GRAND e com a implementação no TopSim, respectivamente. A Tabela 5.3 exibe a quantidade de nós e barras de cada modelo utilizado. A Tabela 5.4 mostra o volume final obtido para cada caso executado. Para o GRAND, foi utilizado nível de conectividade com valor 7 e, para este trabalho, raio de conectividade igual a 7.5.

As estruturas finais possuem as distribuições das barras semelhantes em todos os casos, com simetria presente em todos os resultados.

	Nós	Barras
GRAND	1069	75900
Método Proposto	1083	76928

Tabela 5.3: Número de nós e barras da malha densa de barras para o Modelo Michell utilizados para comparação do resultado da otimização entre o GRAND e o proposto neste trabalho

Geração Malha Densa/Otimização	Volume
GRAND/GRAND	16.2192
Método Proposto/GRAND	16.1836
Método Proposto/Método Proposto	16.1836

Tabela 5.4: Volume obtido na otimização para cada malha densa de barras e o otimizador utilizado

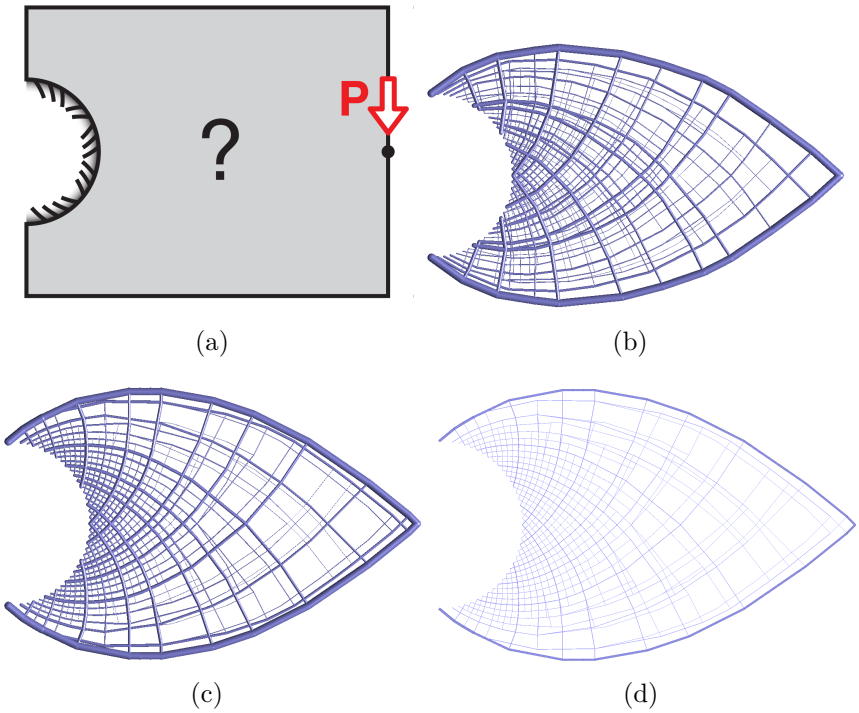


Figura 5.2: Resultado da otimização topológica do Modelo Michell. (a) Geometria, forças aplicadas e condições de contorno. (b) Resultado obtido com a malha densa de barras gerada pelo GRAND, também utilizando o GRAND como otimizador. (c) Resultado obtido com a malha densa de barras gerada pela proposta deste trabalho, utilizando o GRAND como otimizador. (d) Resultado obtido com a malha densa de barras gerada pela proposta deste trabalho, utilizando o otimizador desta dissertação.

5.1.3 Modelo Gancho

O modelo Gancho foi analisado por Talischi e outros (33), com o domínio sendo apresentado na Figura 5.3(a). A Figura 5.3(b) apresenta o resultado com a geração de malha e otimização sendo executadas pelo GRAND e as Figuras 5.3(c) e 5.3(d) mostram os resultados com a geração de malha densa desta dissertação, com o otimizador do GRAND e com a implementação no TopSim, respectivamente. A Tabela 5.5 exibe a quantidade de nós e barras de cada modelo utilizado. A Tabela 5.6 mostra o volume final obtido para cada caso executado. Para o GRAND, foi utilizado nível de conectividade com valor 10 e, para este trabalho, raio de conectividade igual a 17.1.

Apesar das estruturas serem semelhantes, o volume obtido pela malha do GRAND é menor que o volume da malha gerada neste trabalho. Por ser um modelo com poucos nós internos, a maior discretização da malha do GRAND pode explicar essa diferença.

	Nós	Barras
GRAND	728	72589
Método Proposto	593	72641

Tabela 5.5: Número de nós e barras da malha densa de barras para o Modelo Gancho utilizados para comparação do resultado da otimização entre o GRAND e o proposto neste trabalho

Geração Malha Densa/Otimização	Volume
GRAND/GRAND	7154.5791
Método Proposto/GRAND	7171.8273
Método Proposto/Método Proposto	7171.8300

Tabela 5.6: Volume obtido na otimização para cada malha densa de barras e o otimizador utilizado

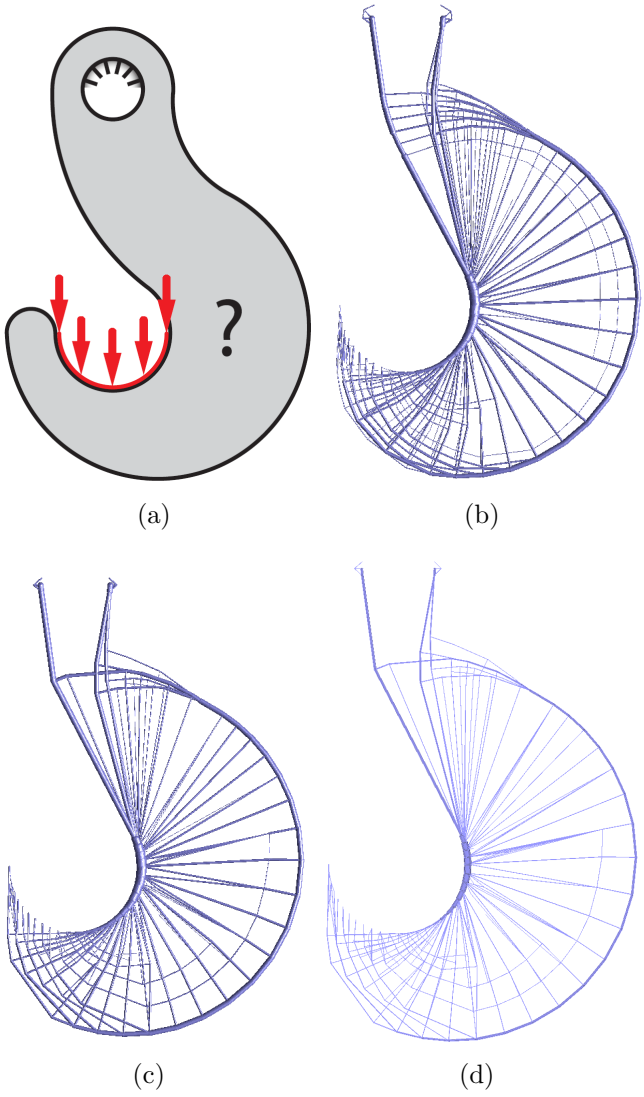


Figura 5.3: Resultado da otimização topológica do Modelo Gancho. (a) Geometria, forças aplicadas e condições de contorno. (b) Resultado obtido com a malha densa de barras gerada pelo GRAND, também utilizando o GRAND como otimizador. (c) Resultado obtido com a malha densa de barras gerada pela proposta deste trabalho, utilizando o GRAND como otimizador. (d) Resultado obtido com a malha densa de barras gerada pela proposta deste trabalho, utilizando o otimizador desta dissertação.

5.1.4 Modelo Serpentine

O modelo Serpentine tem seu domínio sendo apresentado na Figura 5.4(a). A Figura 5.4(b) apresenta o resultado com a geração de malha e otimização sendo executadas pelo GRAND e as Figuras 5.4(c) e 5.4(d) mostram os resultados com a geração de malha densa desta dissertação, com o otimizador do GRAND e com a implementação no TopSim, respectivamente. A Tabela 5.7 exibe a quantidade de nós e barras de cada modelo utilizado. A Tabela 5.8 mostra o volume final obtido para cada caso executado. Para o GRAND, foi utilizado nível de conectividade com valor 7 e, para este trabalho, raio de conectividade igual a 8.3. As estruturas finais possuem as distribuições das barras semelhantes em todos os casos.

	Nós	Barras
GRAND	1045	84706
Método Proposto	920	84488

Tabela 5.7: Número de nós e barras da malha densa de barras para o Modelo Serpentine utilizados para comparação do resultado da otimização entre o GRAND e o proposto neste trabalho

Geração Malha Densa/Otimização	Volume
GRAND/GRAND	91.4121
Método Proposto/GRAND	91.4419
Método Proposto/Método Proposto	91.4419

Tabela 5.8: Volume obtido na otimização para cada malha densa de barras e o otimizador utilizado

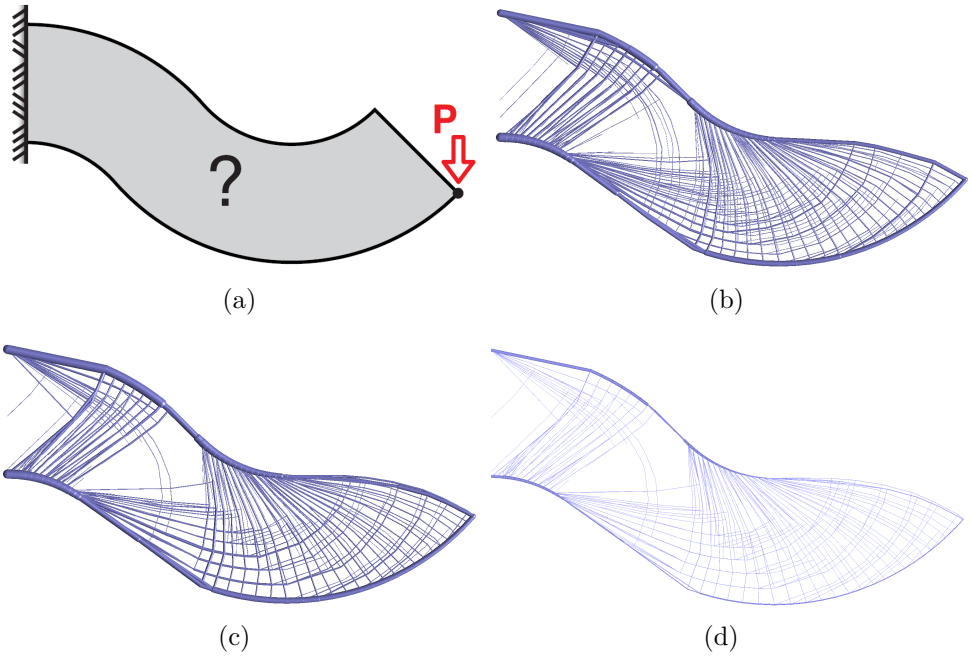


Figura 5.4: Resultado da otimização topológica do Modelo Serpentine. (a) Geometria, forças aplicadas e condições de contorno. (b) Resultado obtido com a malha densa de barras gerada pelo GRAND, também utilizando o GRAND como otimizador. (c) Resultado obtido com a malha densa de barras gerada pela proposta deste trabalho, utilizando o GRAND como otimizador. (d) Resultado obtido com a malha densa de barras gerada pela proposta deste trabalho, utilizando o otimizador desta dissertação.

5.1.5 Modelo Cone

O modelo Cone tem seu domínio sendo apresentado na Figura 5.5(a). A Figura 5.5(b) apresenta o resultado com a geração de malha e otimização sendo executadas pelo GRAND e as Figuras 5.5(c) e 5.5(d) mostram os resultados com a geração de malha densa desta dissertação, com o otimizador do GRAND e com a implementação no TopSim, respectivamente. A Tabela 5.9 exibe a quantidade de nós e barras de cada modelo utilizado. A Tabela 5.10 mostra o volume final obtido para cada caso executado. Para o GRAND, foi utilizado nível de conectividade com valor 3 e, para este trabalho, raio de conectividade igual a 3.3.

Os modelos obtidos em todos os casos tiveram os mesmos valores de volume. Além disso, o valor ótimo do volume dessa estrutura é 16.8076 (6), valor que tende a ser encontrado discretizando o domínio de entrada.

	Nós	Barras
GRAND	1010	115789
Método Proposto	1346	117716

Tabela 5.9: Número de nós e barras da malha densa de barras para o Modelo Cone utilizados para comparação do resultado da otimização entre o GRAND e o proposto neste trabalho

Geração Malha Densa/Otimização	Volume
GRAND/GRAND	17.0310
Método Proposto/GRAND	17.0310
Método Proposto/Método Proposto	17.0310

Tabela 5.10: Volume obtido na otimização para cada malha densa de barras e o otimizador utilizado

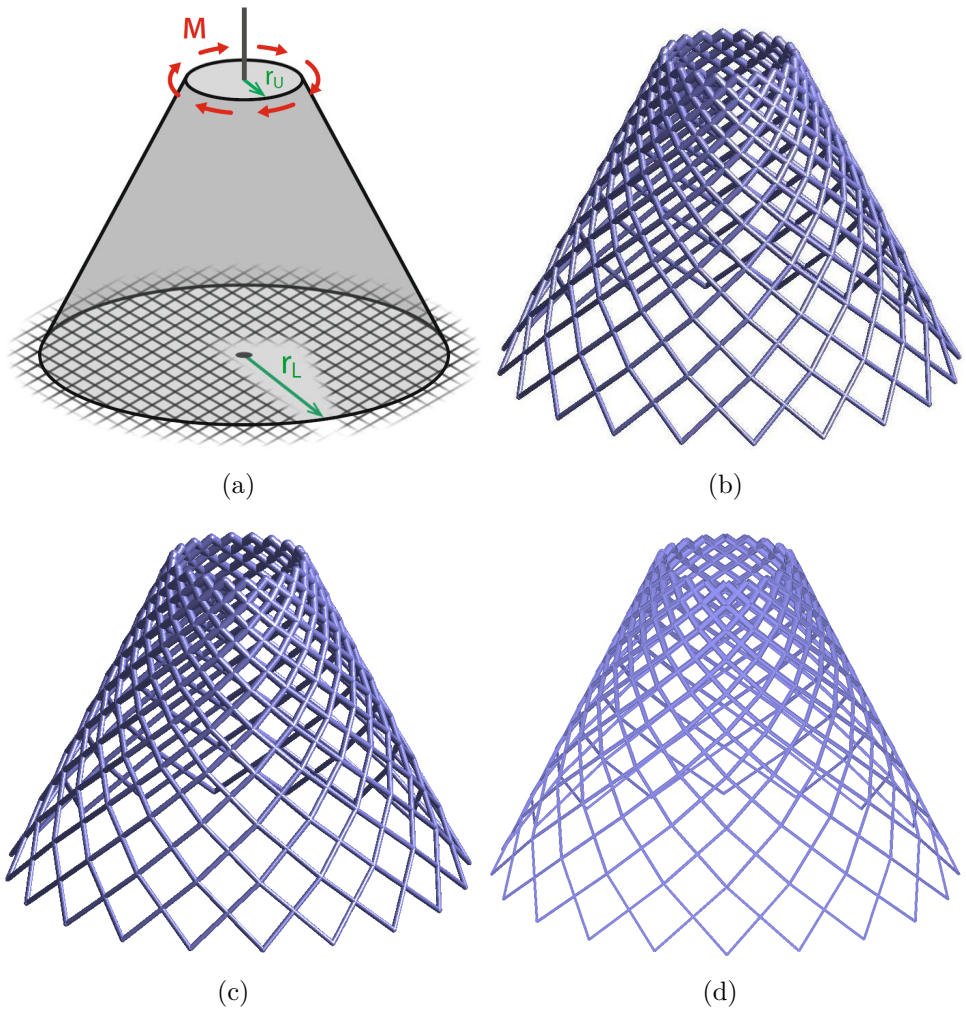


Figura 5.5: Resultado da otimização topológica do Modelo Cone. (a) Geometria, forças aplicadas e condições de contorno. (b) Resultado obtido com a malha densa de barras gerada pelo GRAND, também utilizando o GRAND como otimizador. (c) Resultado obtido com a malha densa de barras gerada pela proposta deste trabalho, utilizando o GRAND como otimizador. (d) Resultado obtido com a malha densa de barras gerada pela proposta deste trabalho, utilizando o otimizador desta dissertação.

5.1.6 Modelo Grua 1

O modelo Grua 1 foi analisado Smith (34, 6), com o domínio sendo apresentado na Figura 5.6(a). A Figura 5.6(b) apresenta o resultado com a geração de malha e otimização sendo executadas pelo GRAND e as Figuras 5.6(c) e 5.6(d) mostram os resultados com a geração de malha densa desta dissertação, com o otimizador do GRAND e com a implementação no TopSim, respectivamente. A Tabela 5.11 exibe a quantidade de nós e barras de cada modelo utilizado. A Tabela 5.12 mostra o volume final obtido para cada caso executado. Para o GRAND, foi utilizado nível de conectividade com valor 5 e, para este trabalho, raio de conectividade igual a 7.

As estruturas obtidas são semelhantes e, apesar do modelo gerado pelo GRAND ser mais discretizado e possuir mais barras, o modelo gerado neste trabalho apresenta um volume menor.

	Nós	Barras
GRAND	935	83922
Método Proposto	872	83705

Tabela 5.11: Número de nós e barras da malha densa de barras para o Modelo Grua 1 utilizados para comparação do resultado da otimização entre o GRAND e o proposto neste trabalho

Geração Malha Densa/Otimização	Volume
GRAND/GRAND	199.5656
Método Proposto/GRAND	193.5080
Método Proposto/Método Proposto	193.5080

Tabela 5.12: Volume obtido na otimização para cada malha densa de barras e o otimizador utilizado

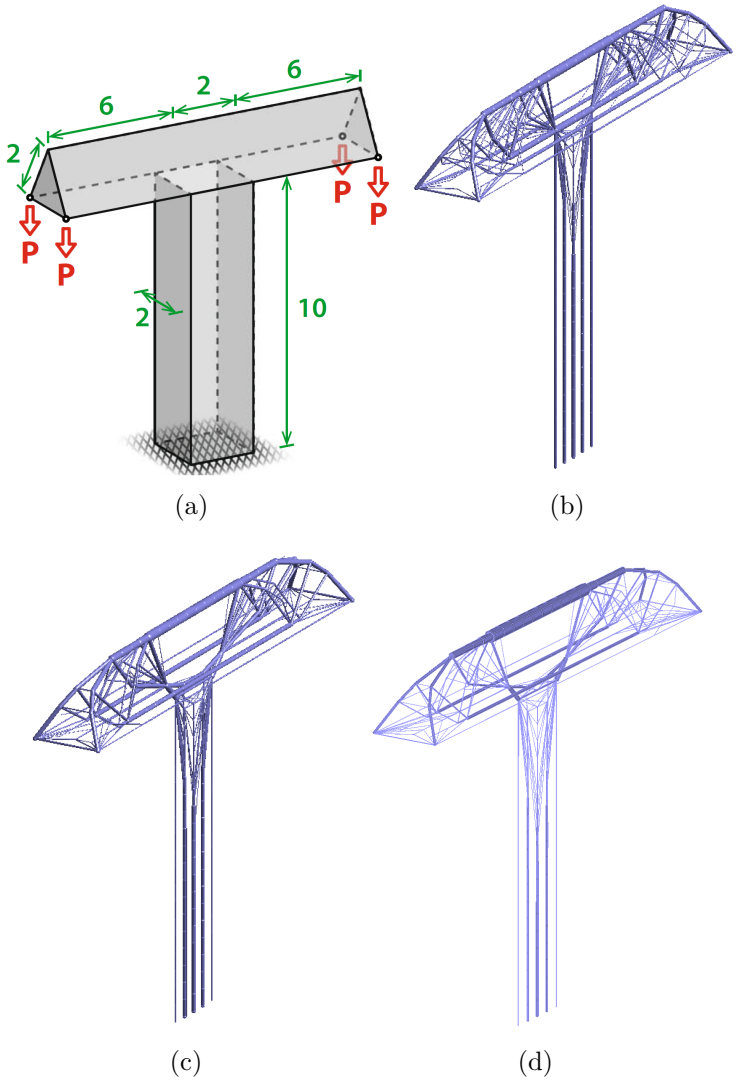


Figura 5.6: Resultado da otimização topológica do Modelo Grua 1. (a) Geometria, forças aplicadas e condições de contorno. (b) Resultado obtido com a malha densa de barras gerada pelo GRAND, também utilizando o GRAND como otimizador. (c) Resultado obtido com a malha densa de barras gerada pela proposta deste trabalho, utilizando o GRAND como otimizador. (d) Resultado obtido com a malha densa de barras gerada pela proposta deste trabalho, utilizando o otimizador desta dissertação.

5.1.7 Modelo Cilindro

O modelo Cilindro tem seu domínio sendo apresentado na Figura 5.7(a). A Figura 5.7(b) apresenta o resultado com a geração de malha e otimização sendo executadas pelo GRAND e as Figuras 5.7(c) e 5.7(d) mostram os resultados com a geração de malha densa desta dissertação, com o otimizador do GRAND e com a implementação no TopSim, respectivamente. A Tabela 5.13 exibe a quantidade de nós e barras de cada modelo utilizado. A Tabela 5.14 mostra o volume final obtido para cada caso executado. Para o GRAND, foi

utilizado nível de conectividade com valor 3 e, para este trabalho, raio de conectividade igual a 4.1.

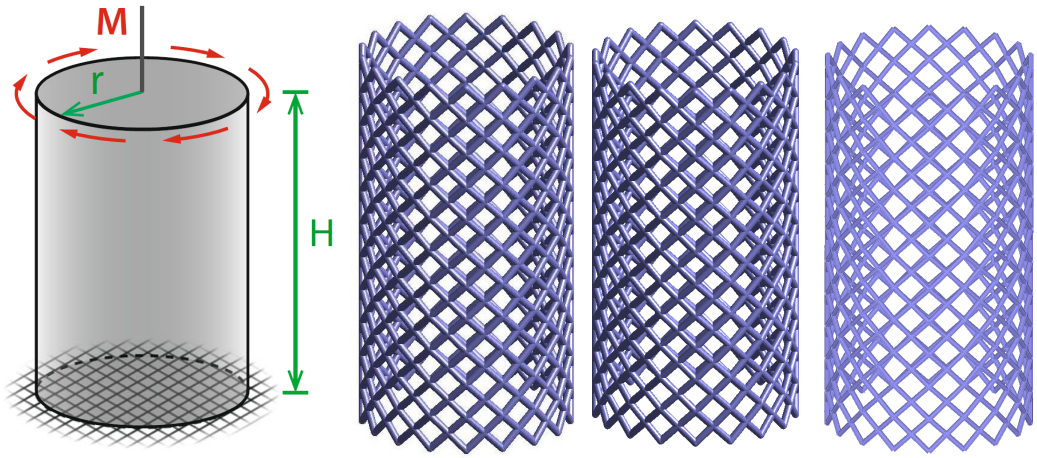
Os modelos obtidos em todos os casos são bastante próximos, somente apresentando uma diferença de 0.0001 na otimização implementada neste trabalho com a otimização do GRAND. Além disso, o valor ótimo do volume dessa estrutura é 36.6667 (6), valor que tende a ser encontrado discretizando o domínio de entrada.

	Nós	Barras
GRAND	1308	152795
Método Proposto	1169	155586

Tabela 5.13: Número de nós e barras da malha densa de barras para o Modelo Cilindro utilizados para comparação do resultado da otimização entre o GRAND e o proposto neste trabalho

Geração Malha Densa/Otimização	Volume
GRAND/GRAND	37.2637
Método Proposto/GRAND	37.2638
Método Proposto/Método Proposto	37.2637

Tabela 5.14: Volume obtido na otimização para cada malha densa de barras e o otimizador utilizado



(a) (b) (c) (d)

Figura 5.7: Resultado da otimização topológica do Modelo Cilindro. (a) Geometria, forças aplicadas e condições de contorno. (b) Resultado obtido com a malha densa de barras gerada pelo GRAND, também utilizando o GRAND como otimizador. (c) Resultado obtido com a malha densa de barras gerada pela proposta deste trabalho, utilizando o GRAND como otimizador. (d) Resultado obtido com a malha densa de barras gerada pela proposta deste trabalho, utilizando o otimizador desta dissertação.

5.2 Desempenho

Nesta seção, é apresentado o desempenho da geração da malha densa de barras e da implementação da otimização topológica estrutural de barras no TopSim. Para os dois casos, será usado o modelo de uma caixa 3D de dimensão $3 \times 1 \times 1$, com as condições de contorno e forças aplicadas ilustradas na Figura 5.8. O modelo utilizado possui 5602 nós iniciais de fronteira, sendo criados 21266 nós internos durante o processo de geração da malha proposto neste trabalho, totalizando 26868 nós. É medido o tempo para a execução de diferentes raios de conectividade (nível de conectividade, no caso do GRAND), tanto para a geração quanto para a otimização. Todos os testes foram realizados numa máquina com 2 processadores Intel Xeon E5-2640 2.40GHz, com 20 *threads* em cada processador, e 256 GB de memória RAM, com o sistema operacional Windows 10. Todo o código deste trabalho foi desenvolvido usando a linguagem C++ e foram utilizados os compiladores do Visual Studio 2012 (VC11) e da Intel C++ Compiler XE 13.0.

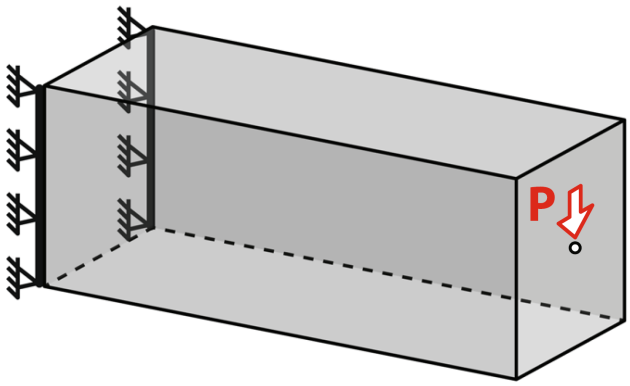


Figura 5.8: Geometria, forças aplicadas e condições de contorno do Modelo de Caixa

5.2.1 Geração da Malha Densa de Barras

A Figura 5.9 exibe a comparação do tempo total para a geração da malha densa de barras entre a técnica proposta nesta dissertação e o GRAND, contabilizando o número de barras potenciais (isto é, o total de barras caso não tivesse o teste de colinearidade) e o número de barras na malha final. Para realizar essa comparação, foi criado um modelo de caixa $3 \times 1 \times 1$ com 24389 nós para ser executado no GRAND.

É importante notar que, mesmo o GRAND partindo de uma malha já previamente discretizada, o método proposto neste trabalho apresenta um

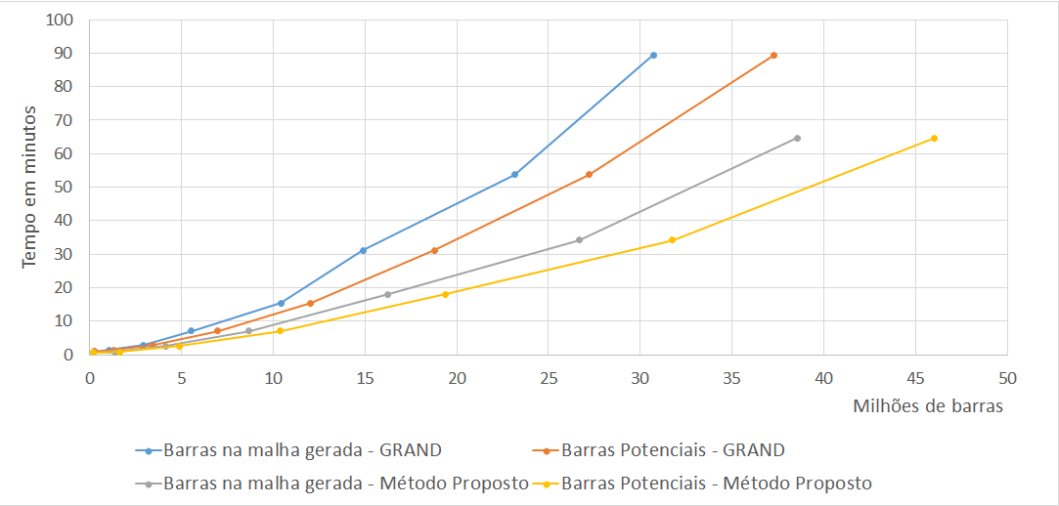


Figura 5.9: Comparação no tempo total de geração da malha densa de barras

desempenho melhor e, quanto maior for o número de barras geradas, melhor será desempenho do método proposto em relação ao GRAND. Por outro lado, o GRAND foi desenvolvido com um foco também educacional, e não somente desempenho, sendo desenvolvido em MATLAB, enquanto todo o trabalho desta dissertação foi feita em C++ e tendo uma maior preocupação com o desempenho do código gerado.

A Figura 5.10 mostra a porcentagem do tempo de processamento das três partes principais do algoritmo proposto, isto é: a classificação das células e o cálculo do campo de distância, a criação dos nós e a geração das barras. Como esperado, quanto maior for o raio de conectividade, maior será o impacto da criação das barras na geração da malha.

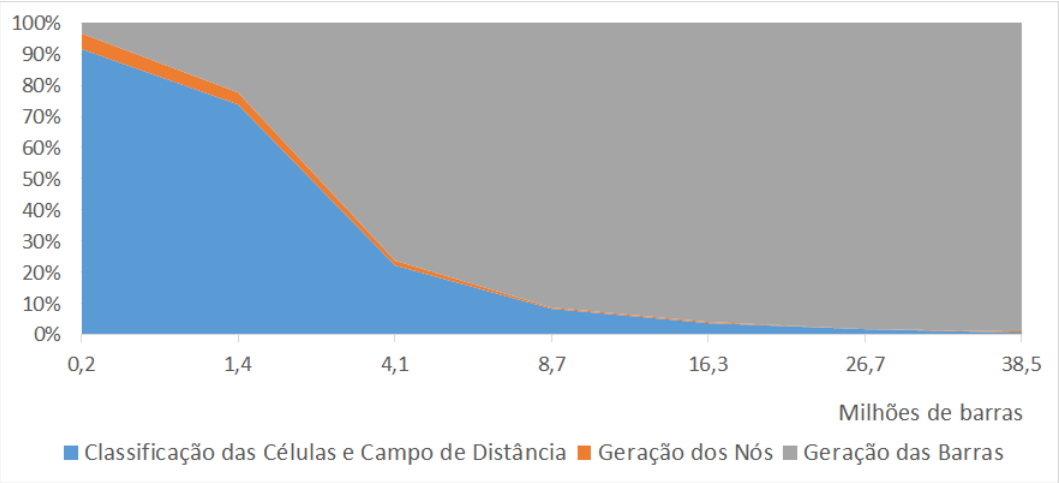


Figura 5.10: Porcentagem no tempo de processamento das partes principais do algoritmo proposto

Com isso, foi computado o tempo de execução das principais partes

do Algoritmo 4 de geração das barras, que são: criação do conjunto com as barras candidatas, sua ordenação e tentativa de adição das barras candidatas na malha. O resultado está presente na Figura 5.11. Com o aumento do número de barras, a parte em que se tenta adicionar as barras na malha passa a ser o gargalo da execução, devido ao teste de colinearidade.

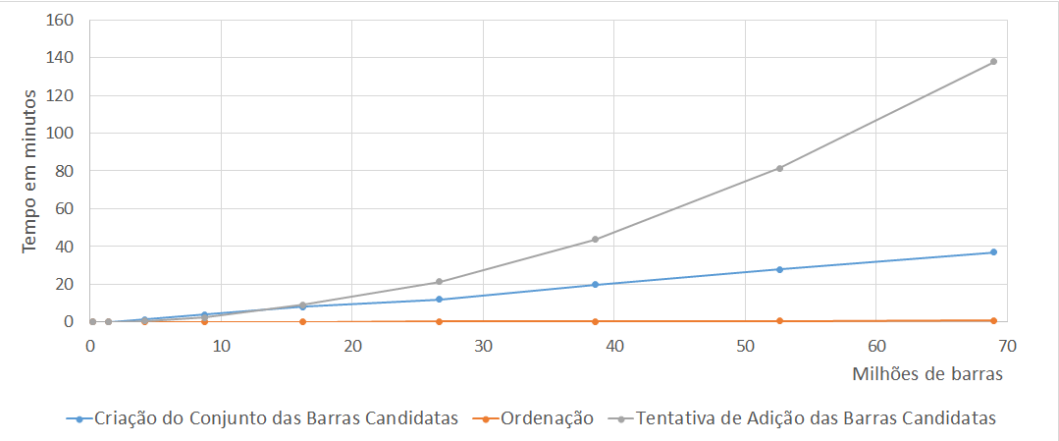


Figura 5.11: Tempo de execução das principais parte da geração das barras

5.2.2 Otimização

A etapa mais custosa da otimização é a resolução do problema de programação linear. A Figura 5.12 exibe o tempo médio da iteração de diferentes níveis de conectividade para o mesmo modelo, com o tempo total da otimização sendo mostrado na Figura 5.13.

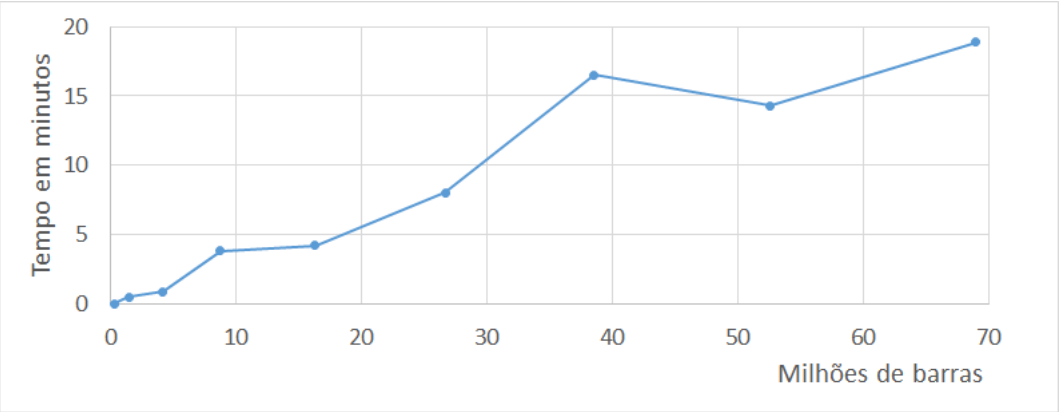


Figura 5.12: Tempo médio das iterações do IPM para diferentes níveis de conectividade

Como mostrado na Figura 5.14, o *solver* é o gargalo da otimização, ocupando quase 80% no tempo total de execução da iteração do IPM. Além disso, devido ao uso da PARDISO, o gasto de memória RAM é o principal

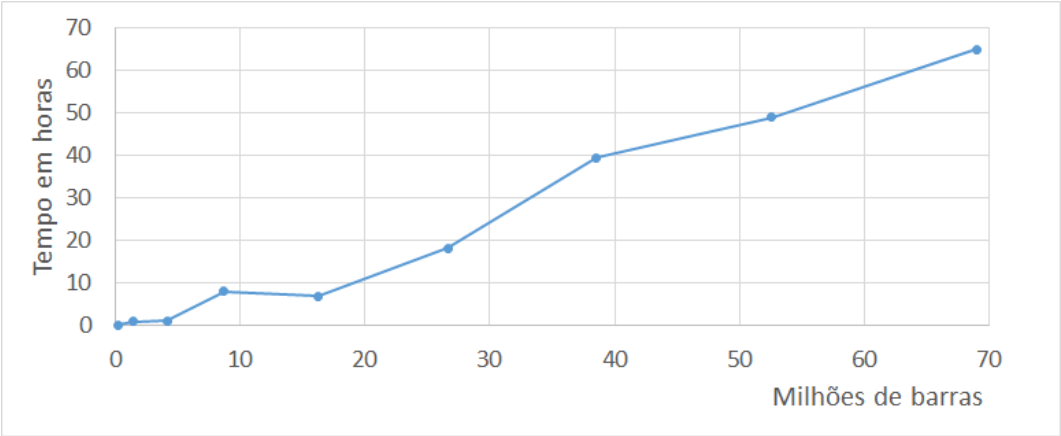


Figura 5.13: Tempo total para resolver o problema de otimização

fator limitante do uso para larga escala, com a Figura 5.15 exibindo o quanto de RAM foi utilizado em cada caso.

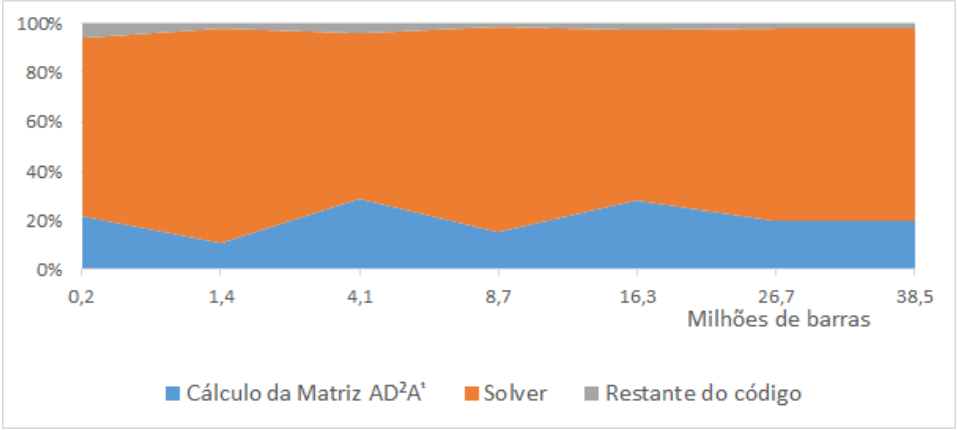


Figura 5.14: Tempo de execução de cada parte da iteração do IPM em porcentagem

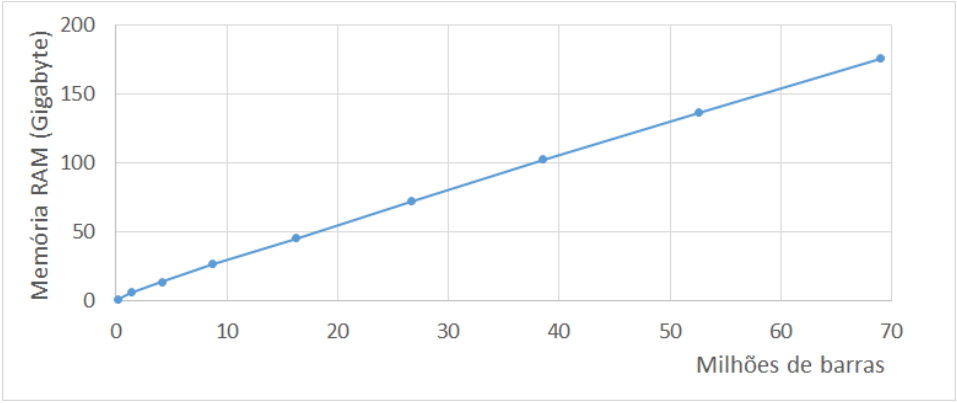


Figura 5.15: Gasto máximo de memória RAM durante a execução da otimização

Por fim, a Figura 5.16 mostra o volume final da estrutura em cada raio de conectividade executado. Como esperado, quanto maior o raio de conectividade, menor o volume final obtido, convergindo para um único valor. A Figura 5.17 exibe o resultado final da otimização com o caso contendo mais de 68 milhões de barras, apresentando duas estruturas que se encontram no ponto de aplicação da força, também como o esperado (35, 6).

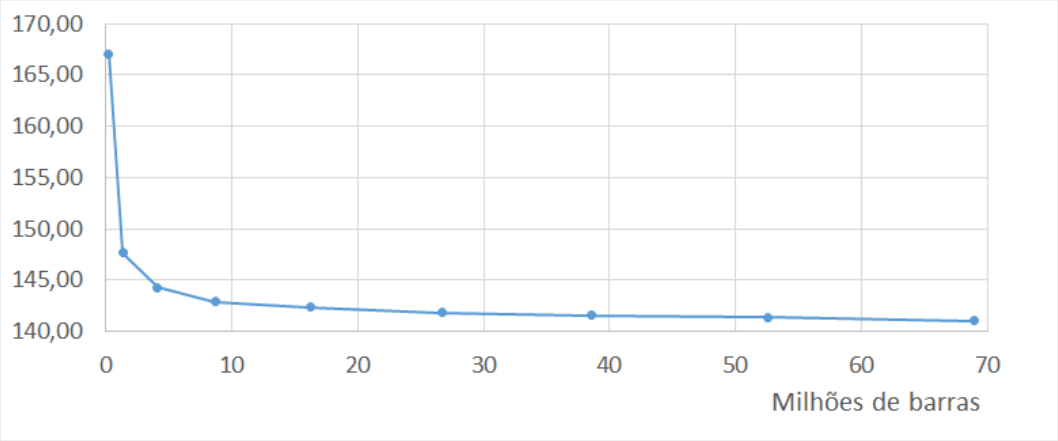


Figura 5.16: Volume obtido para cada otimização realizada

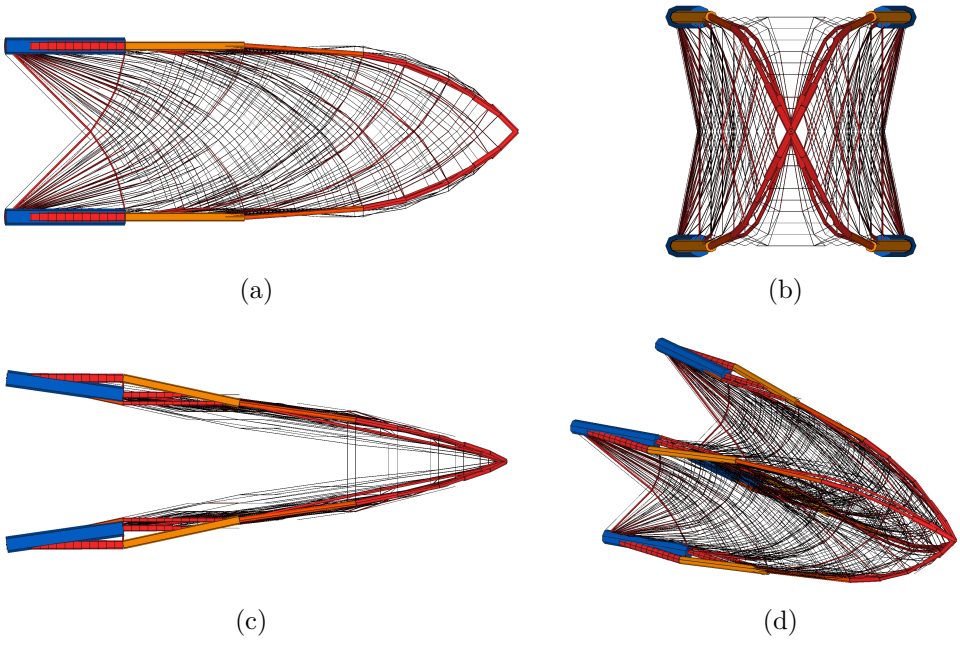


Figura 5.17: Resultado da otimização topológica do Modelo Caixa no caso de 68 milhões de barras. (a) Vista lateral do resultado obtido. (b) Vista frontal do resultado obtido. (c) Vista superior do resultado obtido. (d) Resultado final da otimização topológica.

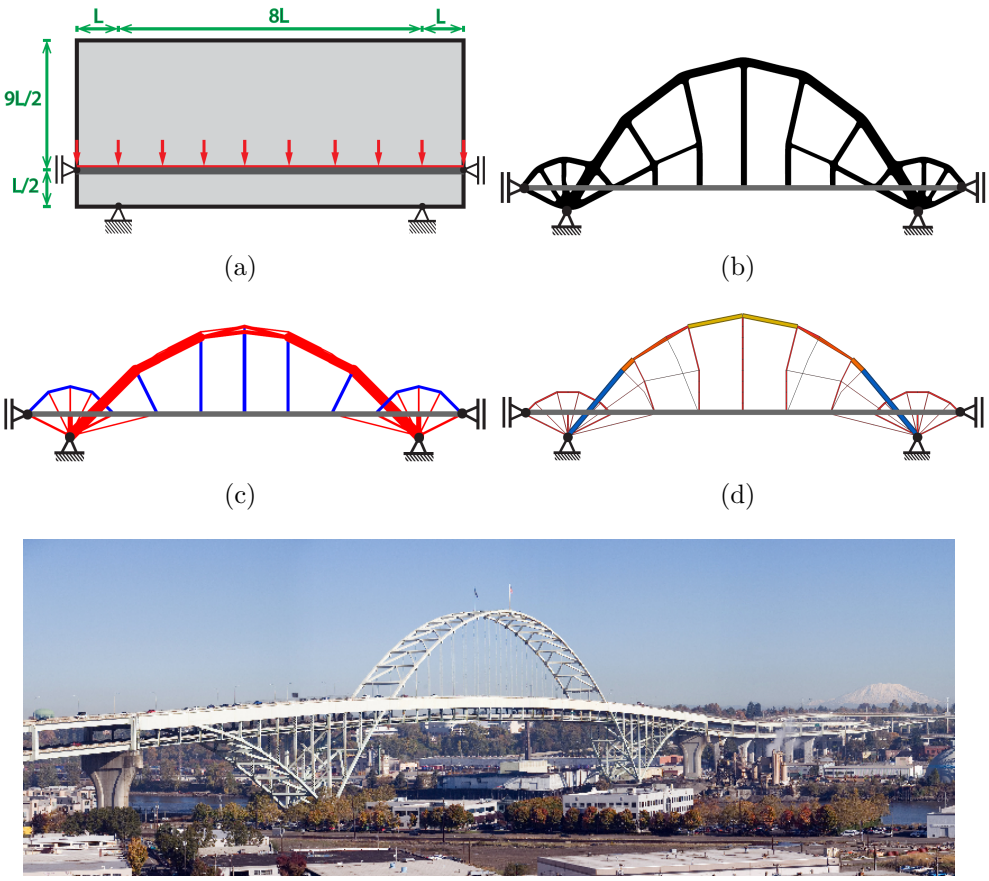
5.3 Outros Modelos

Esta seção apresenta resultados para modelos diferentes dos utilizados na Seção 5.1. Os modelos mostrados nesta seção visam apresentar casos com potencial aplicação real.

5.3.1 Modelo Ponte em Arco

O modelo Ponte em Arco foi analisado por Zhang e outros (7), com o domínio sendo apresentado na Figura 5.18(a). A Figura 5.18(b) exibe o resultado obtido utilizando a otimização topológica contínua, com a Figura 5.18(c) exibindo o resultado da otimização topológica discreta usando a geração de malha densa de barras feita por Zhang e outros e a Figura 5.18(d) mostra o resultado obtido neste trabalho, sendo neste caso utilizado o valor de $cutoff = 0.05$. A Figura 5.18(e) exibe uma ponte real com a mesma forma proposta de arco.

Todos os resultado apresentam formas bem semelhantes da ponte, com as barras obtidas nesta dissertação tendo uma forma um pouco mais próxima ao resultado contínuo do que as barras apresentadas por Zhang e outros.



(a) (b) (c) (d) (e)

Figura 5.18: Resultado da otimização do Modelo Ponte em Arco. (a) Geometria, forças aplicadas e condições de contorno [Retirado de (7)]. (b) Resultado obtido utilizando a otimização topológica contínua [Retirado de (7)]. (c) Resultado obtido em (7). (d) Resultado obtido utilizando a geração de malha densa de barras e a otimização propostas neste trabalho. (e) Fremont Bridge, ponte em arco localizada em Portland, EUA [Retirado de `https://commons.wikimedia.org/wiki/File:FremontBridgePano.jpg`].

5.3.2 Modelo Grua 2

O modelo Grua 2 foi analisado por Liu e outros (31), sendo o domínio apresentado na Figura 5.19(a), com a força do nó 3 sendo igual à soma das forças dos nós 1 e 2. A Figura 5.19(b) mostra o resultado obtido neste trabalho, sendo uma estrutura de boa qualidade para o domínio especificado.

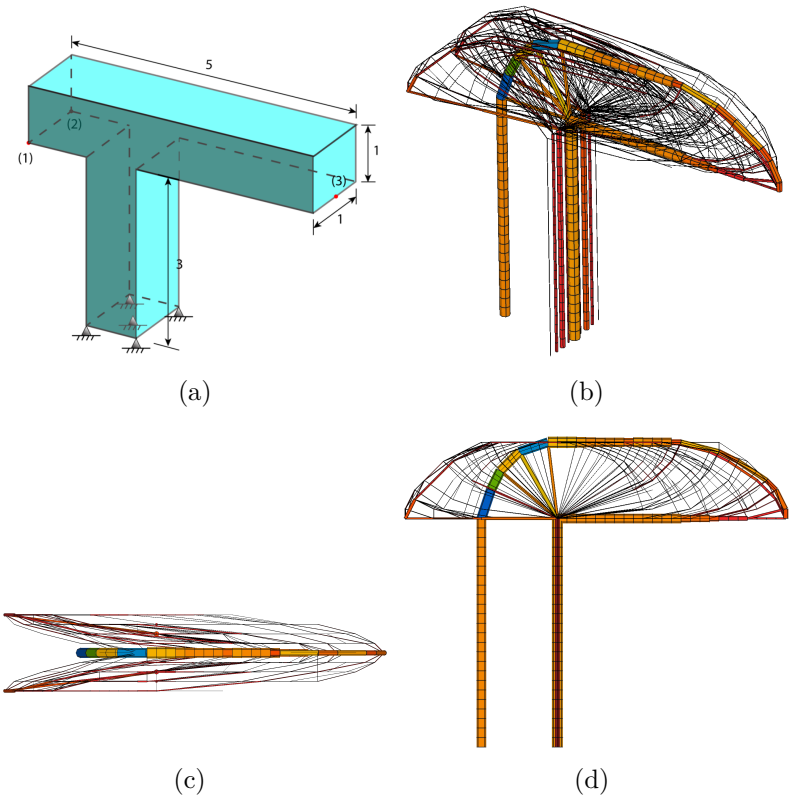


Figura 5.19: Resultado da otimização do Modelo Grua 2. (a) Geometria, forças aplicadas e condições de contorno [Retirado de (31)]. (b) Resultado obtido utilizando a geração de malha densa de barras e a otimização propostas neste trabalho. (c) Vista superior do resultado obtido nesta dissertação. (d) Vista lateral do resultado obtido nesta dissertação.

5.3.3 Modelo Torre

O modelo Torre foi analisado por Zegard e Paulino (32), com o domínio sendo apresentado na Figura 5.20(a). A Figura 5.20(b) exibe o resultado obtido por Zegard e Paulino e a Figura 5.20(c) mostra o resultado obtido neste trabalho. O resultado desta dissertação é compatível com o esperado.

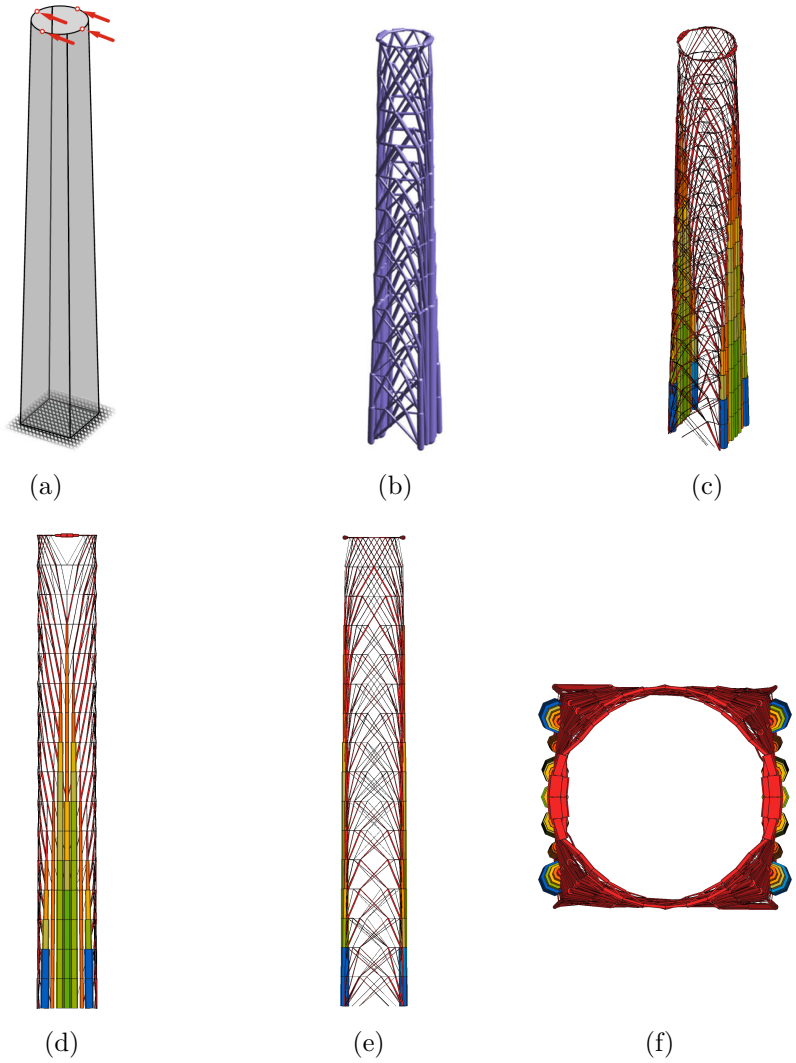


Figura 5.20: Resultado da otimização do Modelo Torre. (a) Geometria, forças aplicadas e condições de contorno [Retirado de (32)]. (b) Resultado obtido em (32).. (c) Resultado obtido utilizando a geração de malha densa de barras e a otimização propostas neste trabalho. (d) Vista lateral do resultado obtido nesta dissertação. (e) Vista frontal do resultado obtido nesta dissertação. (f) Vista superior do resultado obtido nesta dissertação.

6
Conclusão

Nesta dissertação, foi apresentado e discutido o processo de otimização topológica estrutural de barras, desde a geração da malha densa de barras até sua otimização. Foi proposta uma nova forma de geração da malha densa, não dependente de um modelo já discretizado, em contraste com os métodos existentes. A partir da fronteira, são criados os nós respeitando as curvas de seu contorno. Para isso, foi proposto calcular o campo de distância do modelo, utilizando a direção do gradiente do campo para guiar a criação dos nós internos. Feito isso, as potenciais barras da malha são ordenadas através de seu tamanho e tenta-se adicioná-las, privilegiando as barras menores. Para evitar que sejam criadas barras fora do domínio, é feito um teste de colisão da barra com a fronteira. Para agilizar esse processo, e como já havia sido criado uma grade regular, utilizou-se um caminhamento de traçado de raio para computar essa colisão, evitando que fosse necessário calcular cada barra contra toda a fronteira. Esse algoritmo proposto mostrou melhores resultado com relação ao desempenho que o GRAND. Em relação à qualidade, ele apresentou resultados similares a outros trabalhos, sem precisar de uma malha de poligonais (isto é, um modelo discretizado) ou de zonas de restrições que delimitariam as regiões de criação das barras.

Com a malha densa de barras criada, foi implementada a otimização dentro do *framework* TopSim. Para isso, foram desenvolvidos os *plugins* GSM, IPM, NFWriterBrick e Bar. Utilizou-se a formulação plástica na otimização e, para a parte de programação linear, foi implementado o método de Pontos Interiores Dual-Primal Corretor-Preditor proposto por Mehrotra (26). Analisou-se o tempo total da otimização para um modelo de caixa $3 \times 1 \times 1$, sendo apresentado resultados com mais de 68 milhões de barras.

O processo de filtragem não foi abordado nesta dissertação, sendo posto como um trabalho futuro para que se tenha um completo processo de otimização topológica estrutural de barras. Outras ideias de trabalhos futuros são:

- Paralelização da adição das barras (Algoritmo 4), já que este é o gargalo da geração de malha densa de barras.
- Uso de outros *solvers* na implementação do IPM, como o proposto por Cubas Rodriguez (8), para que a memória não seja mais o fator limitante

na otimização.

– Suporte aos casos 2.5D, isto é, superfície 2D imersa em uma dimensão 3D.

Referências bibliográficas

[1] MICHELL, A.. **M. The limits of economy of material in frame structure [J]**. Philosophical Magazine, 8(6):589–597, 1904.

[2] DORN, W. S.. **Automatic design of optimal structures**. Journal de mecanique, 3:25–52, 1964.

[3] CHENG, G. D.; GUO, X.. **ϵ-relaxed approach in structural topology optimization**. Structural optimization, 13(4):258–266, 1997.

[4] DUARTE, L. S.. **TopSim: A plugin-based framework for large-scale numerical analysis**. D.sc. in computer science, Departamento de Informática, PUC-Rio, 2016.

[5] ZEGARD, T.; PAULINO, G. H.. **GRAND — Ground structure based topology optimization for arbitrary 2D domains using MATLAB**. Structural and Multidisciplinary Optimization, 50(5):861–882, 2014.

[6] ZEGARD, T.; PAULINO, G. H.. **GRAND3 — Ground structure based topology optimization for arbitrary 3D domains using MATLAB**. Structural and Multidisciplinary Optimization, 52(6):1161–1184, 2015.

[7] ZHANG, X.; MAHESHWARI, S.; RAMOS JR., A. S. ; PAULINO, G. H.. **Macroelement and Macropatch Approaches to Structural Topology Optimization Using the Ground Structure Method**. Journal of Structural Engineering, 142(11), 2016.

[8] CUBAS RODRIGUEZ, A. E.. **Toward GPU-based ground structures for large scale topology optimization**. M.sc., Departamento de Mecânica, PUC-Rio, 2014.

[9] HEMP, W.. **Optimum structures**. Oxford engineering science series. Clarendon Press, 1973.

[10] OHSAKI, M.. **Optimization of Finite Dimensional Structures**. CRC Press, 2016.

[11] SOKÓŁ, T.. **A 99 line code for discretized Michell truss optimization written in Mathematica**. Structural and Multidisciplinary Optimization, 43(2):181–190, 2011.

[12] ACHTZIGER, W.. **On simultaneous optimization of truss geometry and topology**. Structural and Multidisciplinary Optimization, 33(4):285–304, 2007.

[13] GILBERT, M.; TYAS, A.. **Layout optimization of large-scale pin-jointed frames**. Engineering Computations, 20(8):1044–1064, 2003.

[14] LEVOY, M.. **Area Flooding Algorithms**. In: SIGGRAPH '81 Two-Dimensional Computer Animation course notes, p. 6–12, Dallas, Texas, 1981. ACM. Com correções feitas em 1982.

[15] AMANATIDES, J.; WOO, A.. **A Fast Voxel Traversal Algorithm for Ray Tracing**. In: Eurographics '87, p. 3–10, 1987.

[16] AKENINE-MÖLLER, T.. **Fast 3D Triangle-box Overlap Testing**. In: ACM SIGGRAPH '05 Courses, New York, NY, USA, 2005. ACM.

[17] LEAL, H. R.. **Operações Booleanas na Modelagem por Pontos**. Master's thesis, Departamento de Informática, PUC-Rio, 2004.

[18] JONES, M. W.; BAERENTZEN, J. A. ; SRAMEK, M.. **3D distance fields: a survey of techniques and applications**. IEEE Transactions on Visualization and Computer Graphics, 12(4):581–599, July 2006.

[19] CELES, W.; PAULINO, G. H. ; ESPINHA, R.. **A compact adjacency-based topological data structure for finite element mesh representation**. International Journal for Numerical Methods in Engineering, 64(11):1529–1556, 2005.

[20] GUENNEBAUD, G.; JACOB, B. ; OTHERS. **Eigen v3**. http://eigen.tuxfamily.org, 2010.

[21] SCHENK, O.; GÄRTNER, K.. **Solving unsymmetric sparse systems of linear equations with PARDISO**. Future Generation Computer Systems, 20(3):475 – 487, 2004. Selected numerical algorithms.

[22] SULLIVAN, F.; DONGARRA, J.. **Guest Editors' Introduction: The Top 10 Algorithms**. Computing in Science & Engineering, 2:22–23, 2000.

[23] DANTZIG, G.. **Linear Programming and Extensions**. Princeton Landmarks in Mathematics and Physics. Princeton University Press, 1963.

[24] KHACHIYAN, L.. **Polynomial algorithms in linear programming** . USSR Computational Mathematics and Mathematical Physics, 20(1):53 – 72, 1980.

[25] KARMARKAR, N.. **A New Polynomial-time Algorithm for Linear Programming**. In: Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing, STOC '84, p. 302–311, New York, NY, USA, 1984. ACM.

[26] MEHROTRA, S.. **On the Implementation of a Primal-Dual Interior Point Method**. SIAM Journal on Optimization, 2(4):575–601, 1992.

[27] WRIGHT, S.. **Primal-Dual Interior-Point Methods**. Society for Industrial and Applied Mathematics, 1997.

[28] WRIGHT, S.; NOCEDAL, J.. **Numerical optimization**. Springer Science, 35:67–68, 1999.

[29] ANDERSEN, E. D.; GONDZIO, J.; MÉSZÁROS, C.; XU, X. ; OTHERS. **Implementation of interior point methods for large scale linear programming**. HEC/Université de Genève, 1996.

[30] STANIMIROVIC, P. S.; STOJKOVIC, N. V.; MOMCILOVIC, B. ; JOVANOVIC, Z.. **Augmented and Normal Equations System in Mehrotra's Primal-dual Algorithm**. Filomat, 2001:285–292, 2001.

[31] LIU, K.; PAULINO, G. H. ; GARDONI, P.. **Reliability-based topology optimization using a new method for sensitivity approximation - application to ground structures**. Structural and Multidisciplinary Optimization, 54(3):553–571, 2016.

[32] ZEGARD, T.; PAULINO, G. H.. **Bridging topology optimization and additive manufacturing**. Structural and Multidisciplinary Optimization, 53(1):175–192, 2016.

[33] TALISCHI, C.; PAULINO, G. H.; PEREIRA, A. ; MENEZES, I. F. M.. **PolyTop: a Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes**. Structural and Multidisciplinary Optimization, 45(3):329–357, 2012.

[34] DA SILVA SMITH, O.. **Generation of ground structures for 2D and 3D design domains**. Engineering Computations, 15(4):462–500, 1998.

[35] LEWIŃSKI, T.; ZHOU, M. ; ROZVANY, G.. **Extended exact solutions for least-weight truss layouts—Part I: Cantilever with a horizontal axis of symmetry**. International Journal of Mechanical Sciences, 36(5):375 – 398, 1994.

[36] **Pos3D**. `http://www.tecgraf.puc-rio.br/pt/software/sw-sigma.html`. Acessado: 21-03-2017.

[37] MENEZES, I.; TILIO, M. ; MIRANDA, A.. **Neutralfile**. `https://web.tecgraf.puc-rio.br/neutralfile/`. Acessado: 21-03-2017.