Geração de Malha Densa de Barras para Otimização Estrutural

Vinícius Gama Tavares, Waldemar Celes
Instituto Tecgraf/PUC-Rio
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rio de Janeiro, Brasil
{gama,celes}@tecgraf.puc-rio.br

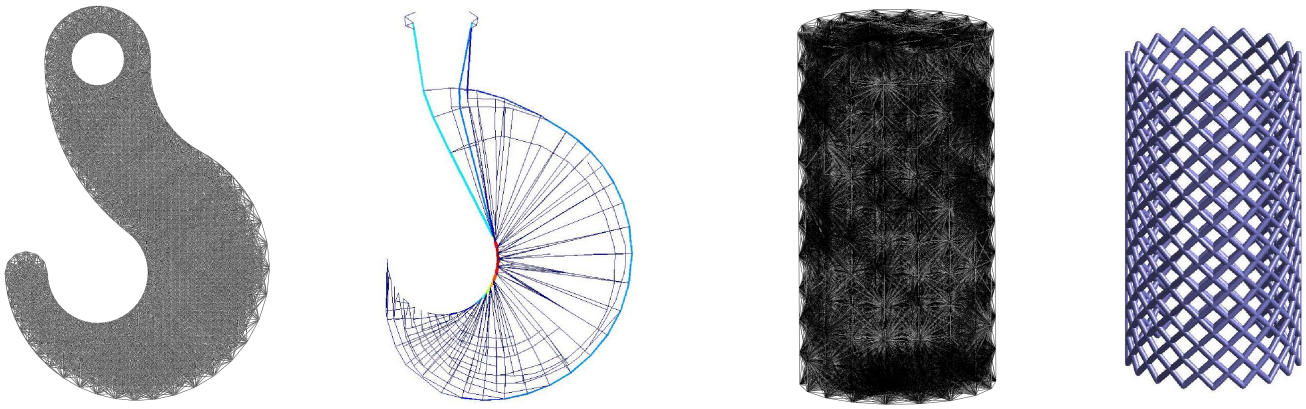


Fig. 1. Malhas densas geradas pelo método proposto e suas respectivas otimizações estruturais: modelo 2D e modelo 3D

***Abstract*—Structural optimization methods are able to identify the optimal truss of a given structure. For that, it is necessary to initially generate a dense bar structure (ground structure, GS), used as input to the optimization method. This work investigates and proposes methods for generating 2D and 3D GS.**

***Keywords*-ground structure; structural optimization; truss;**

***Resumo*—Métodos de otimização estrutural conseguem identificar a treliça ótima de uma determinada estrutura. Para tanto, é necessário gerar inicialmente uma estrutura densa de barras (*ground structure*, GS), usada como entrada do método de otimização. Este trabalho investiga e propõe métodos de geração de GS em 2D e 3D.**

***Palavras-chave*-malha densa de barra; otimização estrutural; treliça;**

I. Introdução

A otimização estrutural topológica 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 identificar a estrutura de treliça ótima de um determinado domínio.

Treliças são sistemas constituídos por elementos indeformáveis, unidos entre si por articulações, consideradas perfeitas, e sujeitos 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 industriais, em torres de transmissão de energia, pontes e diversas outras estruturas. Alguns métodos de otimização estrutural topológica podem ser encontradas em [1].

Esses métodos possuem como entrada uma estrutura densa de barras (*ground structure*), que é definida como uma grade que inclui todos os suportes, potenciais juntas, e conexões que cobrem um domínio [2].Essa estrutura pode ser vista como o preenchimento por barras conectadas através de nós de um determinado domínio.

A contribuição deste trabalho é o desenvolvimento de métodos para a geração das estruturas densas de barras para os casos 2D e 3D. Para a validação desses métodos, as malhas geradas são otimizadas utilizando o software *GRAND* [3] e os resultados são comparados pelos obtidos a partir de malhas densas geradas a partir de uma discretização inicial do domínio.

II. GRAND

A geração de malhas densas no software *GRAND - GRound structure ANalysis and Design*, será usada como inspiração para o desenvolvimento deste trabalho. O *GRAND* realiza tanto a geração da malha densa de barras quanto a identificação das barras para a otimização estrutural. Ele foi desenvolvido em *MATLAB* e foi feito para domínios não estruturados, não ortogonais e côncavos.

Para gerar a estrutura densa de barras, o *GRAND* espera receber uma malha de poligonais, que pode ser uma malha 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 poligonais/poliedro) 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 a nececessidade de uma estrutura topológica para representação das adjacênciais 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.

O método proposto neste trabalho utiliza uma simples estrutura de 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 discretamente a malha densa de barras.

III. Algoritmo Proposto

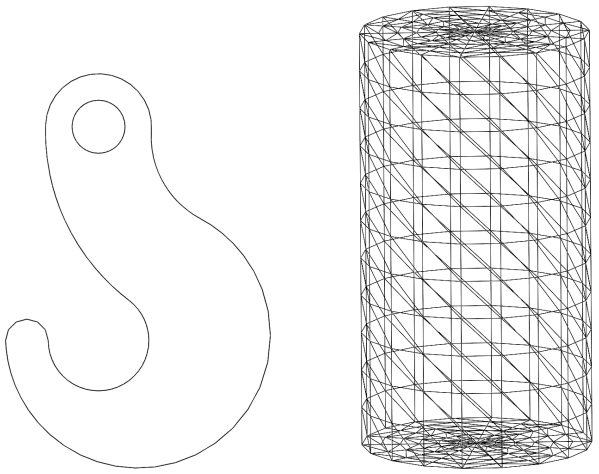
A. Entrada

O algoritmo proposto recebe como entrada o contorno do domínio que se deseja gerar a malha densa de barras, isto é, são necessários somente os pontos que definem a fronteira e as conexões entre eles. Também são recebidos quais nós são fixos e quais nós possuem forças sendo aplicadas. Para a geração da *ground structure* essas informações não são necessárias, porém é preciso para a posterior otimização estrutural.

O domínio pode ser côncavo ou convexo, sendo que, dentro dele, podem existir "buracos". Buracos são regiões dentro do domínio que não permitem a existência de barras atravessando-os. Exemplos de entrada para este trabalho podem ser vistos na Fig. 2.

B. Criação da Grade Regular

Neste trabalho, o domínio será coberto por uma grade regular, que é uma subdivisão da caixa envolvente do domínio em retângulos, paralelogramas ou cuboides congruentes. Uma caixa envolvente é um cuboide (ou um retângulo no caso 2D) que contém todo o domínio. O passo inicial da criação da grade é a definição de qual será o tamanho da célula. Esse valor fica a cargo do usuário, mas para que ele não precise ter conhecimento prévio da malha de entrada, é calculada a distância média entre os pontos conectados da malha. Dessa forma, o usuário não precisa informar o tamanho em valor absoluto da célula, mas sim quantas vezes maior ou menor a célula será em relação ao valor médio das distâncias. Com isso, o tamanho do lado l da célula será $l = (\frac{\sum_{i=1}^{n} dist_i}{n}) * f$, em que n é o número de segmentos de retas da malha de entrada, $dist_i$ é a distância entre dois pontos conectados (segmentos de reta em 2D, arestas em 3D) e f é o fator de multiplicação passado pelo usuário. A escolha do valor de f implicará também no número de nós e, consequentemente, no número de barras que serão geradas, pois quanto menor a célula, maior será o número de barras criadas.



(a) Conjunto de arestas delimitando um domínio 2D

(b) Conjunto de triângulos delimitando um domínio 3D

Fig. 2. Exemplo de Entrada

Tendo o usuário definido o tamanho da célula, a caixa envolvente do domínio é então subdividida. Para evitar problemas de precisão, o tamanho dela é acrescido em 2.5% da extensão do domínio. A escolha de uma grade regular deve-se pela facilidade que existe para descobrir em qual célula um ponto qualquer se encontra. Essa facilidade visa acelerar o processo como um todo da geração de barras. Cada ponto pode ser definido por sua coordenada cartesiana (x, y, z). Para descobrir a qual célula o ponto pertence, calculamos:

$$(i, j, k) = (\lfloor \frac{p_x - g_{x\_min\_grid}}{l_x} \rfloor, \lfloor \frac{p_y - g_{y\_min\_grid}}{l_y} \rfloor, \lfloor \frac{p_z - g_{z\_min\_grid}}{l_z} \rfloor) \quad (1)$$

Em que:

- (i, j, k) é o índice da célula.
- p é o ponto que se deseja saber a qual célula pertence.
- g_{min} é o menor valor de posição da grade.
- l é o tamanho do lado da célula.

C. Classificação das Células e Geração dos nós

Com a criação da grade, podemos classificar as células. São criadas quatro possíveis tipos da célula:

- Células indefinidas, isto é, ainda não foram classificadas.
- Células fora do domínio.
- Células dentro do domínio.
- Células da fronteira do domínio. Nesse caso, é necessária uma especialização:
  - Célula que contém um vértice da entrada.
  - Célula que não contém um vértice da entrada.

Inicialmente, todas as células são classificadas como indefinidas. Para a classificação das células, este trabalho se inspirou em técnicas de preenchimento como o *Flood fill* [4]. Primeiro, classificamos as células do contorno e, com a fronteira já delimitada, classificamos as células restantes. Por exemplo, com todas as células de contorno já definidas, uma célula não classificada, vizinha a outra que já foi classificada como dentro do domínio, também estará dentro do domínio. Para a classificação das células de fronteira, foi necessário implementar algoritmos distintos para o caso 2D e para o caso 3D.

Classificação da Fronteira em 2D: Para cada segmento de reta presente na entrada do domínio, calculamos sua normal através da fórmula $\vec{n} = (-(p1_y - p0_y), (p1_x - p0_x))$, em que $\vec{p0}$ e $\vec{p1}$ são os pontos do segmento de reta. Com isso, cada ponto possui duas normais, uma para cada segmento adjacente. Para cada ponto da entrada, encontra-se a célula a qual ele pertence, através da Equação (1), e por fim associamos esse ponto e suas normais a essa célula. Ela então é classificada como célula de fronteira que contém um vértice da entrada.

Para fechar o contorno, é preciso descobrir quais células os segmentos de reta interceptam. Para isso, foi implementada uma técnica de *Ray Tracing* proposta por Amanatides e Woo [5].

Para cada ponto p da entrada, percorre-se o traçado de raio dele até o ponto ao qual está conectado. Para iniciar o traçado, primeiro deslocamos p até a borda da célula. A partir desse momento, o algoritmo proposto por Amanatides e Woo indica qual a próxima borda interceptada pelo segmento de reta. Com isso, a cada iteração do traçado, percorre-se exatamente uma célula. Isso permite que seja possível saber por qual borda o raio "entrou" e por qual ele "saiu". Então, para essa célula que está na fronteira e não possui um ponto especificado na entrada, geramos um ponto através da média dos pontos descobertos na entrada e na saída do raio e associamos esse ponto, mais a normal da reta, a essa célula. Por fim, ela será classificada como uma célula de fronteira que não contém um vértice da entrada.

Classificação da Fronteira em 3D: O método usado para 3D é análogo ao caso 2D. Para cada triângulo presente na malha inicial, calculamos sua normal através da fórmula $\vec{n} = (\vec{p1} - \vec{p0}) \times (\vec{p2} - \vec{p0})$, em que $\vec{p0}$, $\vec{p1}$ e $\vec{p2}$ são os pontos do triângulo. Para cada ponto da entrada, encontramos a célula a qual ele pertence, através da Equação (1), e por fim associamos esse ponto e suas normais a essa célula. Ela então é classificada como célula de fronteira que contém um vértice da entrada.

Como no 2D, para fechar o contorno é preciso descobrir quais células que os triângulos da entrada interceptam. Para isso, foi implementada a técnica de colisão entre um triângulo e uma caixa proposta por Akenine-Möller [6]. O algoritmo apresenta os seguintes passos:

- Calcula-se a caixa envolvente do triângulo que deseja descobrir quais células ele intercepta.
- Com a caixa envolvente, encontra-se quais são as células canditadas a colidirem com o triângulo.
- Para cada célula canditada, 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.

Se uma célula de fronteira é encontrada, então precisamos gerar um ponto para ser associado a ela. Para isso, foi implementado o algoritmo *Sutherland-Hodgman Clipping* [7], pois com ele é possível descobrir pontos na borda da célula que colidem com o triângulo. Analogamente ao caso 2D, é feita então a média desses pontos e esse novo ponto encontrado é associado a célula, junto com a normal do triângulo.

Classificação das Células Restantes: Enquanto as células de fronteira vão sendo classificadas, elas também são postas num conjunto. Terminada a classificação de todas as células de contorno, para a classificação das células que restaram, o seguinte algoritmo é feito:

while $!set.empty()$ **do**
  $cell \leftarrow set.get()$
  $neighbors \leftarrow get\_neighbors(cell)$
  for all $neighbors$ **do**
    if $neighbor.type() == undefined$ **then**
      $classify(neighbor)$
      if $neighbor.type() == domain\_in$ **then**
        $set.put(neighbor)$
      end if
    end if
  end for
end while

Em que a função $get\_neighbors(cell)$ retorna uma lista todas as células vizinhas, isto é, todas as células que compartilham pelo menos um vértice com ela. Por exemplo, em 2D, os vizinhos da célula $(1, 1)$ são: $(0, 0); (1, 0); (2, 0); (0, 1); (2, 1); (0, 2); (1, 2); (2, 2)$.

Se o vizinho c ainda não foi classificado, então ele é 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 fronteiras e/ou células indefinidas, então é necessário que c passe por um teste. Esse teste é definido através da seguinte regra [8]:

> 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:
>
> $$(x - p) \cdot n_p > 0 \Leftrightarrow x \notin V. \quad (2)$$

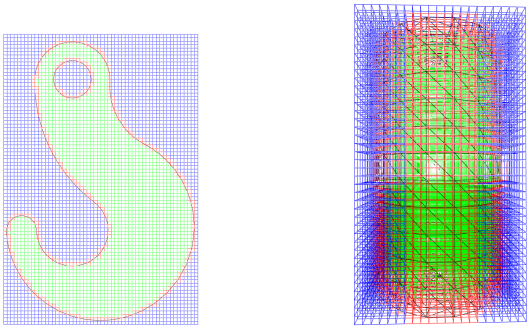
Isto é, essa regra visa determinar em qual parte da superfície formada pelo ponto do contorno e pela normal, o vetor entre os pontos está, podendo assim definir se o ponto x está dentro ou fora do domínio.

Com essa regra, os vértices v_i que compõe c são analisados através da Equação (2), comparando-os aos mais próximos nós que pertencem a células de fronteira. Este é o motivo pelo qual

todas as células de contorno necessitam possuir pelo menos um ponto e uma normal. Então, para todo v_i é encontrado o nó d mais próximo a ele. Encontrado d, v_i é classificado com todas as normais pertencentes a d através da Equação (2). Após todos os v_i serem classificados, a célula a será classificada com a mesma classificação de todos os seus vértices. Se a célula possuir vértices com classificações diferentes, então ela não é classificada neste momento e será classificada quando um dos seus vizinhos for classificado.

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. Por fim, as células que estão dentro do domínio geram um ponto dentro de si, para ser usado futuramente como nó das barras. Esse ponto pode ser gerado centralizado ou randomicamente deslocado a partir do centro da célula. Por isso, quanto menor o tamanho da célula, maior será a quantidade de nós gerados, já que cada célula dentro do domínio irá criar um nó.

O resultado da classificação das células pode ser visto na Fig. 3.



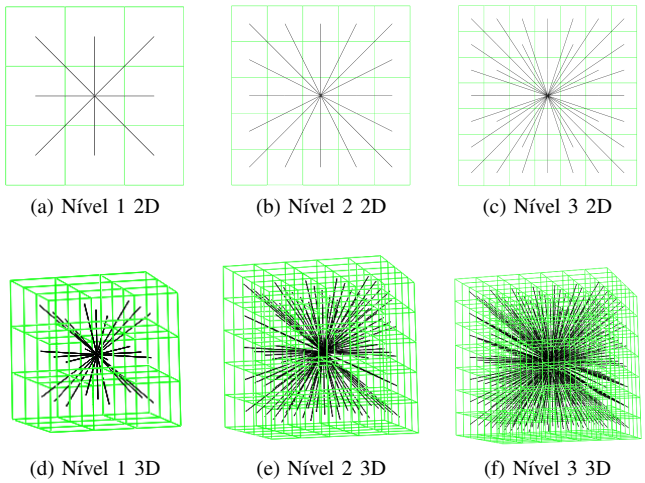
(a) 2D (b) 3D

Fig. 3. Resultado da Classificação das Células. Células azuis representam células de fora do domínio, verdes representam células dentro do domínio e vermelhas as células de fronteira

D. Geração das Barras

No processo de geração das barras, é necessária a especificação de dois parâmetros: o nível de conectividade e o ângulo de tolerância para a colinearidade, os mesmos definidos pelo *GRAND*. O nível de conectividade representa o quão distante uma barra pode conectar dois nós. Por exemplo, o nível de conectividade 1 representa tentar ligar o nó de uma célula aos nós das suas células vizinhas diretas, o nível 2 representa tentar ligar aos nós das vizinhas de suas células vizinhas e assim por diante. Na Fig. 4, é possível ver como funciona o nível de conectividade.

As barras são geradas a partir de nós de células válidas, que são células de dentro do domínio e células de fronteira que contém um vértice da entrada. Não são geradas barras a partir de nós pertencentes a células de fronteira sem vértices



(a) Nível 1 2D (b) Nível 2 2D (c) Nível 3 2D

(d) Nível 1 3D (e) Nível 2 3D (f) Nível 3 3D

Fig. 4. Níveis de conectividade

da entrada para não alterar a fronteira do domínio. Por isso há a necessidade da especificação das células de fronteira. Por não conterem nós, também não são geradas barras a partir de células de fora do domínio.

Ao tentar adicionar uma barra, todos os seguintes testes são feitos:

- Se o destino da barra é válido.
- Se há alguma barra colienar.
- Se a barra atravessa uma parte fora do domínio.

A barra só será adicionada se passar por todos esses testes. Se o destino da barra for uma célula não-válida, isto é, uma célula de fronteira sem vértice de entrada ou uma célula fora do domínio, a barra é descartada. Já o teste de colinearidade existe para não permitir que barras colineares sejam criadas, pois isso dificulta a futura otimização estrutural. Ao tentar adicionar uma barra, é verificado o ângulo entre a nova barra e todas as outras barras do ponto origem e do ponto destino. Se um desses ângulos for menor que o valor especificado pelo ângulo de tolerância para a colinearidade, essa barra não é adicionada.

Se a barra cruzar a fronteira do domínio, então ela precisa ser descartada. Para isso, é feito um traçado de raio do início da barra até o final dela. Para cada célula que a barra intercepta, são feitas as seguintes verificações:

- Se for uma célula de dentro do domínio, continua o traçado
- Se for uma célula de fora do domínio, descarta a barra
- Se for uma célula de fronteira, verifica se há colisão

No caso 2D, a colisão é entre um segmento de reta com outros dois segmentos de reta. Já no caso 3D, a colisão é entre segmento de reta e triângulos. Por isso, para cada caso, são feitos algoritmos diferentes para detectar a colisão e eles são explicados a seguir.

Colisão segmento de reta com segmento de reta: Um segmento de reta pode ser escrito na forma:

$$\vec{P_a} = \vec{A_1} + u_a(\vec{A_2} - \vec{A_1})$$

em que $\vec{A_1}$ e $\vec{A_2}$ são pontos do segmento. Definindo outro segmento de reta $\vec{P_b} = \vec{B_1} + u_b(\vec{B_2} - \vec{B_1})$, deseja-se resolver a equação $\vec{P_a} = \vec{P_b}$. Resolvendo a equação temos:

$$u_a = \frac{(x_{B2} - x_{B1})(y_{A1} - y_{B1}) - (y_{B2} - y_{B1})(x_{A1} - x_{B1})}{(y_{B2} - y_{B1})(x_{A2} - x_{A1}) - (x_{B2} - x_{B1})(y_{A2} - y_{A1})}$$

$$u_b = \frac{(x_{A2} - x_{A1})(y_{A1} - y_{B1}) - (y_{A2} - y_{A1})(x_{A1} - x_{B1})}{(y_{B2} - y_{B1})(x_{A2} - x_{A1}) - (x_{B2} - x_{B1})(y_{A2} - y_{A1})}$$

Se $0 < u_a < 1$ e $0 < u_b < 1$, então os segmentos de reta se interceptam entre os pontos $\vec{A_1}$ e $\vec{A_2}$ e entre $\vec{B_1}$ e $\vec{B_2}$. Com isso, houve uma colisão entre a barra e fronteira, sendo a barra descartada.

Colisão segmento de reta com triângulo: Foi implementado um algoritmo proposto por Dan Sunday [9], inspirado em [10]. Ele executa os seguintes passos:

- Verifica se o segmento de reta intercepta o plano do triângulo
- Calcula o ponto de intercessão entre o segmento de reta e o plano do triângulo
- Verifica se este ponto está dentro do triângulo, através da fórmula paramétrica do triângulo $\vec{V}(s,t) = \vec{V_0} + s(\vec{V_1} - \vec{V_0}) + t(\vec{V_2} - \vec{V_0})$, em que $\vec{V_0}$, $\vec{V_1}$ e $\vec{V_2}$ são os vértices do triângulo e s e t são números reais tais que:
  - $0 \leq s$
  - $0 \leq t$
  - $s + t \leq 1$

Com isso, é preciso resolver a equação:

$$s_I = \frac{(\vec{u} \cdot \vec{v})(\vec{w} \cdot \vec{v}) - (\vec{v} \cdot \vec{v})(\vec{w} \cdot \vec{u})}{(\vec{u} \cdot \vec{v})^2 - (\vec{u} \cdot \vec{u})(\vec{v} \cdot \vec{v})}$$

$$t_I = \frac{(\vec{u} \cdot \vec{v})(\vec{w} \cdot \vec{u}) - (\vec{u} \cdot \vec{u})(\vec{w} \cdot \vec{v})}{(\vec{u} \cdot \vec{v})^2 - (\vec{u} \cdot \vec{u})(\vec{v} \cdot \vec{v})}$$

Em que $\vec{u} = \vec{V_1} - \vec{V_0}$, $\vec{v} = \vec{V_2} - \vec{V_0}$ e $\vec{w} = \vec{P_i} - \vec{V_0}$, em que $\vec{P_i}$ é o ponto calculado entre a intercessão do segmento de reta e o plano.

Se $0 \leq s_i$, $0 \leq t_i$ e $s_i + t_i \leq 1$, então o ponto está dentro do triângulo e a barra colidiu com a fronteira, sendo esta barra descartada.

Resultado da detecção da colisão: Após tratados todos os casos em que a barra intercepta uma célula de fronteira, somente as barras que não colidem com a fronteira são mantidas. Os resultados da geração de barras são ilustrados na Fig. 1.

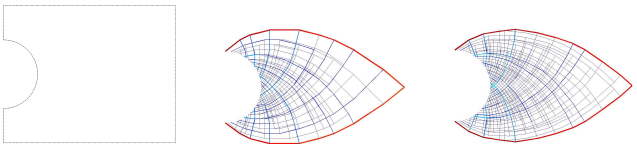
IV. Resultados

Para validação das malhas densas geradas, são comparados os resultados obtidos pela otimização estrutural do *GRAND* a partir da malha densa geradas pelo próprio *GRAND* e do nosso método. Os valores de colinearidade e de nível de conectividade serão os mesmos e também os valores de nós serão os mais próximos possíveis. Isto é, como definido em III-B, o valor de f será tal que o número de nós criados seja próximo ao número de nós da malha gerada pelo *GRAND*.

A Tabela I exibe o número de nós e barras geradas pelo nosso método e pelo *GRAND* e da Fig. 5 até a Fig. 11 são comparados os resultados da otimização a partir de malhas geradas pelo algoritmo proposto e pelo algoritmo de geração do *GRAND*. Nos resultados dos modelos 2D as cores representam barras com o mesmo valor da área de secção transversal.

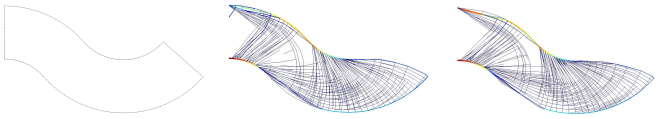
Tabela I
Comparação do número de nós e número de barras

Modelo	Número de Nós		Número de Barras	
	GRAND	Proposto	*GRAND*	Proposto
Michell 2D (Fig. 5)	1069	1046	28256	23480
Serpentine 2D (Fig. 6)	1045	1049	49831	48692
Gancho 2D (Fig. 7)	728	807	72589	82270
Flor 2D (Fig. 8)	2100	2072	69400	72241
Grua 3D (Fig. 9)	1069	1046	47076	48881
Cilindro 3D (Fig. 10)	1308	1333	152795	130415
Cone 3D (Fig. 11)	1010	1028	115789	95664



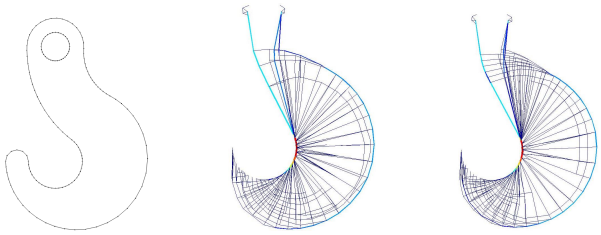
(a) (b) (c)

Fig. 5. Resultados da otimização estrutural do modelo Michell: (a) Modelo de entrada. (b) Resultado do método proposto. (c) Resultado do GRAND



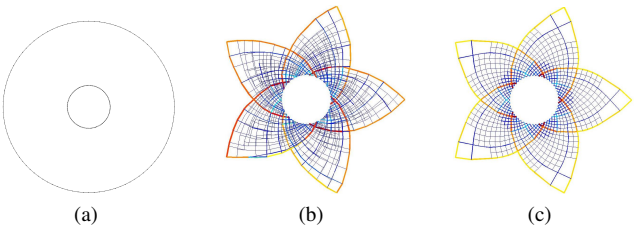
(a) (b) (c)

Fig. 6. Resultados da otimização estrutural do modelo Serpentine: (a) Modelo de entrada. (b) Resultado do método proposto. (c) Resultado do GRAND



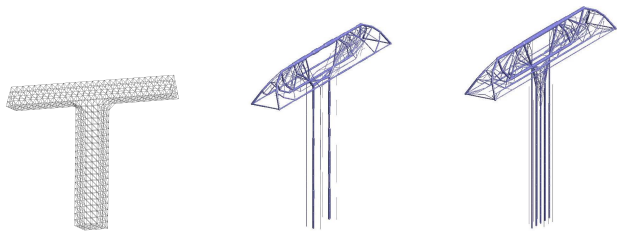
(a) (b) (c)

Fig. 7. Resultados da otimização estrutural do modelo Gancho: (a) Modelo de entrada. (b) Resultado do método proposto. (c) Resultado do GRAND



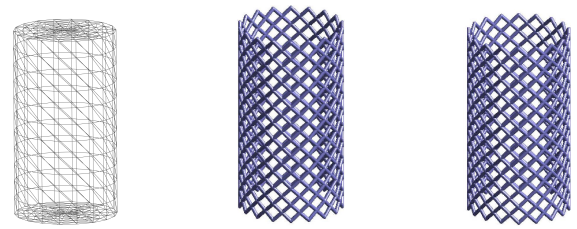
(a) (b) (c)

Fig. 8. Resultados da otimização estrutural do modelo Flor: (a) Modelo de entrada. (b) Resultado do método proposto. (c) Resultado do GRAND



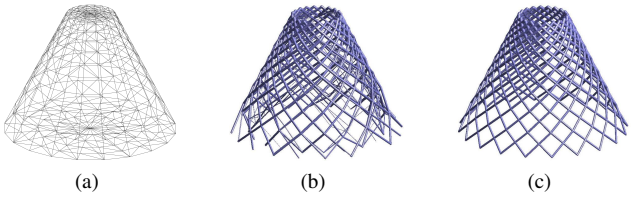
(a) (b) (c)

Fig. 9. Resultados da otimização estrutural do modelo Grua: (a) Modelo de entrada. (b) Resultado do método proposto. (c) Resultado do GRAND



(a) (b) (c)

Fig. 10. Resultados da otimização estrutural do modelo Cilindro: (a) Modelo de entrada. (b) Resultado do método proposto. (c) Resultado do GRAND



(a) (b) (c)

Fig. 11. Resultados da otimização estrutural do modelo Cone: (a) Modelo de entrada. (b) Resultado do método proposto. (c) Resultado do GRAND

Tempo de Geração da Malha: Para a validação do tempo de geração da malha densa de barras, foram geradas novas malhas para entrada da Fig. 10a. A Fig. 12 representa o custo em tempo da geração da malha a partir da variação do número de nós. Neste caso, o nível de conectividade foi fixado com valor 3 e o número de nós variou de 1418 até 416876. A Fig. 13 representa o custo da variação do nível de conectividade. Neste caso, foi mantido o número de nós fixo igual a 10478 e variado o nível de conectividade de 1 até 10.

Fig. 12. Variando o número de nós com o nível de conectividade fixo

V. Discussão e Conclusão

Este trabalho teve como objetivo criar um novo método para a geração de malhas densas de barras a partir do contorno de uma malha. A qualidade da otimização usando malhas geradas pelo nosso método é similar aos resultados usando malhas geradas pelo *GRAND*, porém os resultados dos modelos Flor (Fig. 8) e Cone (Fig. 11) mostram que o software *GRAND* gera malhas de melhor qualidade, visto os resultados da otimização estrutural que são obtidos. Porém, como explicado na Seção II, o custo para a geração dessas malhas é muito alto, já que é necessário que o domínio esteja discretizado e também tenha definido suas zonas de restrições.

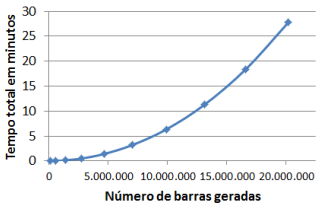


Fig. 13. Variando o nível de conectividade com o número de nós fixo

Estudando as malhas de poligonais usadas como entrada para o *GRAND*, vimos um alinhamento entre os nós dos polígonos, gerando assim barras já alinhadas com as fronteiras, e isso não ocorre em nosso método, já que não existe nenhum tipo de alinhamento entre os nós de entrada e os nós gerados. Com isso, surge a oportunidade de um novo trabalho futuro de usar campo de distância para guiar a criação preferencial de barras nas direções perpendiculares e paralelas em relação a fronteira.

Com relação ao desempenho, vemos na Fig. 12 que o tempo da geração da malha é linearmente proporcional ao número de barras criadas. Já na Fig. 13 vemos um crescimento exponencial do tempo, mas isso ocorre devido ao problema proposto, que é variar o nível de conectividade da malha. Quanto maior o nível, maior o número de verificações que uma barra precisa passar para ser adicionada à malha, resultando em um crescimento não linear.

Referências

[1] M. Bendsøe, A. Ben-Tal, and J. Zowe, "Optimization methods for truss geometry and topology design," *Structural optimization*, vol. 7, no. 3, pp. 141–159, 1994.

[2] W. Dorn, R. Gomory, and H. Greenberg, "Automatic design of optimal structures," *Journal de mecanique*, vol. 3, pp. 25–52, 1964.

[3] T. Zegard and G. Paulino, "Grand — ground structure based topology optimization for arbitrary 2d domains using matlab," *Structural and Multidisciplinary Optimization*, vol. 50, no. 5, pp. 861–882, 2014.

[4] M. Levoy, "Area flooding algorithms," in *SIGGRAPH '81 Two-Dimensional Computer Animation course notes*. Dallas, Texas: ACM, 1981, pp. 6–12, Com correções feitas em 1982.

[5] J. Amanatides and A. Woo, "A fast voxel traversal algorithm for ray tracing," in *Eurographics '87*, 1987, pp. 3–10.

[6] T. Akenine-Möller, "Fast 3d triangle-box overlap testing," in *ACM SIGGRAPH '05 Courses*. New York, NY, USA: ACM, 2005.

[7] A. Vardy, disponível em: http://www.cs.mun.ca/av/old/teaching/cg/notes/raster_clip2_inclass.pdf.

[8] H. R. Leal, "Operações Booleanas na Modelagem por Pontos," Master's thesis, Departamento de Informática, PUC-Rio, 2004.

[9] D. Sunday, disponível em: http://geomalgorithms.com/a06-_intersect-2.html.

[10] T. Möller and B. Trumbore, "Fast, minimum storage ray-triangle intersection," *J. Graph. Tools*, vol. 2, no. 1, pp. 21–28, Oct. 1997.