A PARALLEL TECHNIQUE FOR TWO-DIMENSIONAL MESH GENERATION FOR ARBITRARY REGIONS WITH CRACKS

Markos Oliveira Freitas[a], Joaquim Bento Cavalcante-Neto[a], Creto Augusto Vidal[a], Luiz Fernando Martha[b], Anthony Ingraffea[c] e Paul Wawrzynek[c]

[a]*CRAb, Computer Graphics, Virtual Reality and Animation Group, Computing Department, Federal University of Ceará, Campus do Pici, Av. Humberto Monte, s/n, 60440-554, Fortaleza, CE, Brazil, crab.dc.ufc.br*

[b]*Tecgraf, Computer Graphics Technology Group, Pontifical Catholic University of Rio de Janeiro, Rua Marquês de São Vicente, 225, Gávea, 22451-900, Rio de Janeiro, RJ, Brazil, www.tecgraf.puc-rio.br*

[c]*CFG, Cornell Fracture Group, Cornell University, Ithaca, NY, United States, www.cfg.cornell.edu*

Palavras Chave: Parallel mesh generation; advancing front technique; recusive decomposition.

Resumo. Este trabalho descreve uma técnica para gerar malhas triangulares bidimensionais com fraturas utilizando computadores paralelos de memória distribuída, baseada no modelo de paralelismo mestre/escravos. Esta técnica utiliza uma *quadtree* pouco refinada para decompor o domínio e um procedimento de avanço de fronteira sequencial para gerar a malha de cada subdomínio concorrentemente. Para avançar a fronteira para um subdomínio vizinho, cada subdomínio sofre um deslocamento em uma direção cartesiana, e o mesmo procedimento de avanço de fronteira é executado no domínio deslocado. Este procedimento de deslocamento/geração de malha é repetidamente aplicado até que não possa gerar mais malha, deslocando os subdomínios cada vez em uma direção diferente. Uma *quadtree* mais refinada também é empregada para ajudar a estimar a carga de processamento associada com cada subdomínio. Esta técnica de estimativa de carga produz resultados que representam fielmente o número de elementos a serem gerados em cada subdomínio, levando a uma boa predição de tempo de execução e a um algoritmo bem balanceado. Quando comparado com sua versão sequencial, a técnica paralela é mais rápida e frequentemente apresenta um desempenho superlinear. As malhas geradas com a técnica paralela apresentam a mesma qualidade que as geradas sequencialmente, dentro de limites aceitáveis.

1 INTRODUÇÃO

Este trabalho apresenta uma técnica paralela para gerar malhas bidimensionais triangulares pelo método de avanço de fronteira. A técnica foi projetada para satisfazer cinco requisitos:

1. Respeitar a fronteira dada como entrada, discretizada em segmentos, isto é, nenhum refinamento da borda pode ser empregado.
2. Produzir elementos de boa qualidade, evitando elementos com razão de aspecto ruins.
3. Proporcionar boas transições entre regiões da malha com grande densidade de elementos e regiões com baixa densidade de elementos da malha.
4. Respeitar fraturas existentes na fronteira de entrada.
5. Gerar malhas eficientemente em termos de tempo de processamento.

O primeiro requisito é muito importante em diversos problemas, tais como aqueles encontrados em simulações nas quais o domínio contém regiões com diferentes materiais ou com furos. Nesses problemas, é desejável que a malha seja compatível com a discretização existente da borda destas regiões.

Com relação ao segundo requisito, embora a técnica proposta não garanta limites na razão de aspecto dos elementos, ela procura gerar os melhores elementos possíveis em cada passo. Na Seção 4, será mostrado que a técnica atende este requisito com sucesso.

Com respeito ao terceiro requisito, em diversas aplicações a diferença de tamanho entre os elementos de uma região com alta densidade de elementos e aqueles de uma região com baixa densidade de elementos é maior que duas ordens de magnitude. Logo, proporcionar uma boa capacidade de transição é um requisito importante na prática.

O quarto requisito é importante em simulações de propagação de fraturas, pois permite uma geração local de malha ao redor da fratura. Nessa malha, a superfície da fratura é representada como uma região de área nula formada por segmentos de borda geometricamente coincidentes, cujas normais têm sentidos opostos.

Para realizar o quinto requisito, a técnica proposta utiliza uma arquitetura computacional de memória distribuída com uma capacidade de balanceamento de carga muito alta. Esse tipo de arquitetura tem sido amplamente disponibilizado hoje em dia com o barateamento de *clusters*. A técnica paralela apresentada neste trabalho para problemas bidimensionais utiliza o paradigma de paralelismo mestre/escravos e é facilmente expansível para três dimensões.

O restante deste trabalho está dividido em quatro seções. A próxima seção descreve os trabalhos relacionados. A Seção 3 descreve a técnica paralela desenvolvida, onde um modelo hipotético bidimensional é utilizado para ilustrar o procedimento. A Seção 4 mostra alguns exemplos, em que são avaliadas medidas importantes de estimativa e balanceamento de carga, de *speed-up* e de qualidade da malha. Finalmente, algumas conclusões são apresentadas na Seção 5.

2 TRABALHOS RELACIONADOS

Chrisochoides (2005) classifica as técnicas paralelas de geração de malha de acordo com a abordagem de decomposição do domínio. As técnicas na primeira abordagem, chamada de decomposição discreta de domínio (DDD), iniciam gerando sequencialmente uma malha pouco refinada para a fronteira de entrada. Então, essa malha pouco refinada é particionada, geralmente com alguma técnica de particionamento de grafos (Karypis e Kumar, 1998; Karypis et al., 2011). A borda externa e as interfaces internas são refinadas, e pode-se gerar malha em cada partição independentemente das outras. Alguns trabalhos da literatura são baseadas em abordagens DDD (Hodgson e Jimack, 1996; Wilson e Topping, 1998;

Chrisochoides e Nave, 2000; Rivara et al., 2006; Ito et al., 2007). Embora alguns desses trabalhos sejam em três dimensões, eles são mencionados aqui por que a mesma ideia pode ser empregada em técnicas bidimensionais.

A outra abordagem apresentada em Chrisochoides (2005) é chamada de decomposição contínua de domínio (DCD). Com essa abordagem, qualquer método diferente do particionamento de uma malha pouco refinada pode ser utilizado para decompor o domínio. Logo, métodos tais como *quadtree*/*octree* (De Cougny e Shephard, 1999; Löhner, 2001; Chernikov e Chrisochoides, 2005), eixos/planos (Lämmer e Burghardt, 2000; Zagaris et al., 2009), blocos (Okusanya e Peraire, 1996; Chernikov e Chrisochoides, 2006) ou particionamento de estruturas de dados (Kohout et al., 2005) estão entre as escolhas possíveis.

O algoritmo descrito neste trabalho utiliza a abordagem DCD e gera malha interfaceando subdomínios *a posteriori* por alguns motivos importantes. Primeiro, não é recomendável criar uma malha pouco refinada como um passo obrigatório precedente à geração de uma malha refinada, já que um refinamento da borda não é permitido devido ao primeiro requisito apresentado na Seção 1; segundo, é desejável particionar a geometria presente na entrada, e uma técnica de particionamento de grafos ou de malhas não é geralmente projetado para este tipo de tarefa; terceiro, como observado em Chrisochoides (2005), uma malha interfaceando dois subdomínios gerada *a priori* poderia levar a artefatos, tais como elementos pobremente formados ou transições severas entre regiões com elementos pequenos e regiões com elementos grandes, invalidando o segundo e o terceiro requisitos.

Em Okusanya e Peraire (1996); Chernikov e Chrisochoides (2005); Kohout et al. (2005); Chernikov e Chrisochoides (2006), uma malha de Delaunay bidimensional é gerada a princípio, ou é dada como entrada e refinada em paralelo. O refinamento é realizado pela inserção paralela de novos pontos em diversas localizações da malha, garantindo que não conflitam, ou seja, os elementos refinados pela inserção de um novo ponto não serão afetados pela inserção simultânea de outros pontos. Para garantir isso, Chernikov e Chrisochoides (2006) particionam a malha dada utilizando blocos de mesmo tamanho e, em Chernikov e Chrisochoides (2005), os mesmos autores propõem um particionamento por uma *quadtree*. Okusanya e Peraire (1996) também empregam particionamento por blocos, mas utilizam comunicação interprocessos para evitar inserções paralelas inválidas. Kohout et al. (2005) utilizam particionamento de estruturas de dados e empregam mecanismos de memória compartilhada para evitar a geração concorrente inválida de novos triângulos.

Em Lämmer e Burghardt (2000), dada uma descrição bidimensional da borda, uma reta coincidindo com o eixo principal de inércia é criado para dividir o domínio. Aplicando este procedimento recursivamente, até que o número de subdomínios se iguale ao número de processadores, arestas são criadas nos eixos internos e na borda, e malhas de Delaunay ou quadrangulares são geradas simultaneamente. Em Zagaris et al. (2009), um plano que divide a borda 3D de entrada é definido. Entretanto, em vez de gerar faces no plano, uma técnica de avanço de fronteira (AFT, *Advancing Front Technique*) é aplicada somente nas faces da borda que interceptam o plano, gerando uma camada de elementos tetraédricos que separa o domínio. Esse procedimento é realizado recursivamente, gerando fronteiras desconexas, que são avançadas paralelamente em diferentes processadores.

Em De Cougny e Shephard (1999) e em Löhner (2001), o domínio de entrada é decomposto por uma *octree* e uma técnica de avanço de fronteira é utilizado nos subdomínios. Em De Cougny e Shephard (1999), a malha interna é gerada pela aplicação de padrões pré-definidos em uma *octree* refinada. Os elementos conectando a borda de entrada e a malha gerada por padrões são criados por uma AFT, enquanto não interceptarem os planos definidos

pelas células da *octree*. Esses dois passos de geração de malha são executados em paralelo. Um passo de comunicação interprocessos de três fases é utilizado para construir uma malha conectando as malhas geradas: primeiro, entre duas células adjacentes (separadas por um plano); então, entre as células que compartilham a mesma aresta (definida por dois planos que se interceptam); e, por último, entre as células que compartilham o mesmo vértice (definido por mais de dois planos que se interceptam).

Em Löhner (2001), somente uma AFT é utilizada para gerar a malha de todo o domínio. Depois de construir uma *octree* pouco refinada, malhas são geradas em cada célula da borda de entrada, contanto que os tetraedros não cruzem a célula. As células sofrem um deslocamento diagonal definido empiricamente para reduzir o tamanho da fronteira, eliminando quase todas as faces entre duas células adjacentes. Então, uma *octree* ainda menos refinada é gerada, e o mesmo procedimento é realizado, até que nenhuma malha possa mais ser gerada.

Embora haja similaridades importantes, o trabalho descrito em Löhner (2001) e o presente trabalho são diferentes em diversos aspectos. Primeiro, todo subdomínio em Löhner (2001) sofre o mesmo deslocamento diagonal fixo enquanto que, no presente trabalho, um deslocamento cartesiano é aplicado e varia de acordo com o tamanho do subdomínio. Segundo, o único propósito do deslolcamento aplicado por Löhner (2001) é o de diminuir o número de faces da fronteira, enquanto que o fator que possibilita que a AFT continue é a criação da *octree* menos refinada. No presente trabalho, o deslocamento de subdomínios tem uma participação maior no procedimento da AFT, já que é ela a responsável pelo avanço da fronteira para um subdomínio vizinho.

Entretanto, a diferença principal entre os dois trabalhos é a técnica de estimativa da carga nos processadores. Em Löhner (2001), a carga é estimada por um particionamento regular do subdomínio. Apesar de essa estimativa de carga representar uma malha uniforme, não é suficientemente precisa por não levar em conta a não-uniformidade da malha gerada e, portanto, pode levar a desbalanceamentos. O presente trabalho resolve esse problema com a utilização de uma *quadtree* refinada, já que ela pode representar tanto regiões com elementos grandes como regiões com pequenos elementos. Além disso, é a estimativa de carga que guia a geração dos subdomínios no presente trabalho, em vez de estimar a carga depois da geração dos subdomínios.

Sobretudo, este artigo diferencia-se dos trabalhos acima mencionados por satisfazer todos os requisitos citados na Seção 1, especialmente no que diz respeito ao tratamento de fraturas.

3 DESCRIÇÃO DA TÉCNICA PARALELA

A presente técnica recebe como entrada uma lista de segmentos definindo a fronteira de um ou mais objetos, que podem ou não conter furos ou fraturas. Essa fronteira e seu interior definem o domínio, que é decomposto por uma *quadtree*, construída de forma a manter a quantidade de trabalho necessária para se gerar malha em suas células-folha abaixo de um limiar especificado. Esses subdomínios, correspondentes às folhas da *quadtree*, são distribuídos entre os processadores escravos. Cada um desses processadores gera malha no subdomínio a ele atribuído.

A fronteira é atualizada e as células da *quadtree* de decomposição são deslocadas em uma direção cartesiana. Os subdomínios deslocados são novamente distribuídos entre os processadores escravos, onde mais malhas são geradas, e a fronteira é novamente atualizada. Esse procedimento é repetido, deslocando-se as células da *quadtree* de decomposição para outras direções, até que não seja mais possível gerar malha. A Figura 1 dá uma visão global da técnica apresentada nesse trabalho.

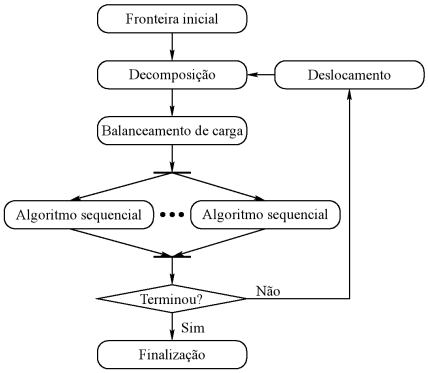

Figura 1: Visão global da técnica paralela.

Depois desse procedimento de geração de malha, executado nos processadores escravos, o processador mestre finaliza a malha, gerando elementos nas regiões remanescentes, e aplicando uma melhoria da qualidade da malha em regiões onde os processadores escravos simplesmente geraram uma malha pouco refinada.

3.1 Estimativa da Carga

Em computação de alto desempenho (CAD), a carga é uma medida da quantidade de trabalho a ser realizado em um subdomínio ou em uma coleção de subdomínios. Em problemas de geração de malha, a carga está relacionada ao número de elementos que serão gerados em cada subdomínio. Portanto, uma boa estimativa de carga deve levar em conta as seguintes questões:

- O nível de discretização da malha, geralmente especificado pelo usuário ou por outro programa de computador, através de parâmetros de entrada. Quanto maior a discretização, maior a carga (Figura 2).
- Em regiões com o mesmo nível de discretização, uma região maior gera mais elementos que uma região menor. Logo, quanto maior a região, maior a carga (esquerda da Figura 3).
- Em domínios com nível de discretização variável, uma região de transição em que os elementos variam de tamanho não deve ser contabilizada como uniforme, pois existirão mais elementos em algumas zonas e menos elementos em outras zonas dentro da mesma região (direita da Figura 3).

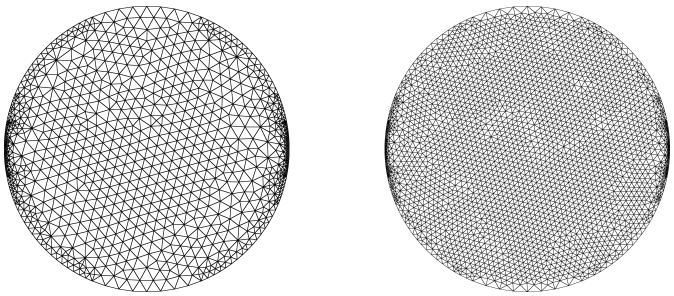

Figura 2: Estimativa de carga: Malha menos refinada (esquerda) e malha mais refinada (direita).

A malha da esquerda da Figura 2 sugere uma carga maior do que a sugerida pela malha da direita da mesma figura. A mesma discretização de borda foi utilizada nos dois casos. Na Figura 3, a carga associada às regiões cercadas por linhas contínuas é maior que a carga associada às regiões cercadas por linhas tracejadas.

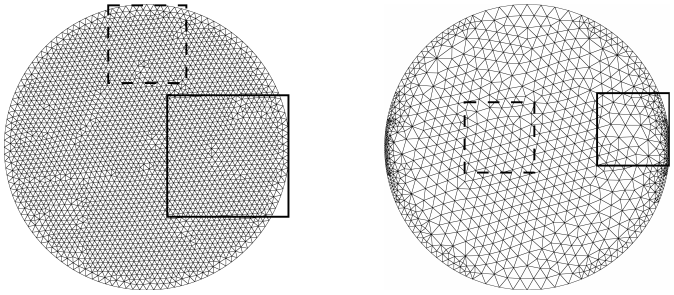

Figura 3: Estimativa de carga: malha uniforme (esquerda) e malha não-uniforme (direita).

Neste trabalho, com o objetivo de obter uma boa estimativa, uma *quadtree* de estimativa de carga é cuidadosamente construída de forma que reflita os tamanhos dos elementos dentro do domínio do objeto. Logo, uma ideia global da malha desejada deve ser conhecida *a priori*. Felizmente, esse conhecimento pode ser inferido das duas restrições impostas na função de tamanho dos elementos da malha. A primeira restrição é que um elemento no interior não deve ser maior do que o maior elemento na borda do objeto. A segunda restrição é que a malha final deve apresentar boas transições entre regiões com diferentes graus de refinamento.

Já que essas restrições estão relacionadas aos tamanhos dos elementos, elas devem estar representadas na *quadtree* de estimativa de carga. Depois de sua construção, a *quadtree* de estimativa de carga inicial passa por dois refinamentos: o primeiro para prevenir que uma célula interna seja maior do que a maior célula da borda, e a segunda para evitar que a diferença de nível entre duas células adjacentes seja maior do que um (este último é conhecido na literatura como refinamento 2:1). Mais detalhes de como essa *quadtree* é construída podem ser encontrados em outros trabalhos anteriores dos autores (Miranda et al., 1999, 2009; Cavalcante-Neto et al., 2001)).

O número de células-folha dessa *quadtree* que não estiverem fora do domínio de entrada, ou seja, o número de folhas dentro do domínio mais as folhas da borda, é considerado como a carga total para esse domínio.

Note que, como somente a carga deve estar representada na *quadtree* de estimativa de carga, seu primeiro refinamento não precisa satisfazer estritamente às duas restrições. Logo, células completamente internas podem ser maiores do que as células de borda. O número de células-folha de uma célula completamente interna C pode ser conhecida previamente, sem ter que refiná-la. Esse número é calculado como $4^{d-d(C)}$, onde d é a profundidade desejada da árvore e $d(C)$ é a profundidade de C. O refinamento 2:1 tem que ser totalmente aplicado, uma vez que ele pode modificar o número de células-folha de C.

Na esquerda da Figura 4, uma *quadtree* de estimativa de carga com os dois refinamentos descritos é ilustrada. As células consideradas para efeito de estimativa de carga são aquelas que se encontram dentro do domínio (células escuras) e aquelas que interceptam a borda do domínio (células preenchidas). Esse tipo de *quadtree* deve ser construído depois de cada deslocamento da *quadtree* de decomposição, e nova estimativa de carga deve ser computada. Isso pode causar uma economia significativa de memória e de tempo de execução, especialmente quando o problema envolve a geração de malhas muito grandes.

3.2 Decomposição do Domínio

O domínio é decomposto por uma outra árvore chamada *quadtree* de decomposição, que é guiada pela *quadtree* de estimativa de carga descrita na Seção 3.1. A *quadtree* de decomposição

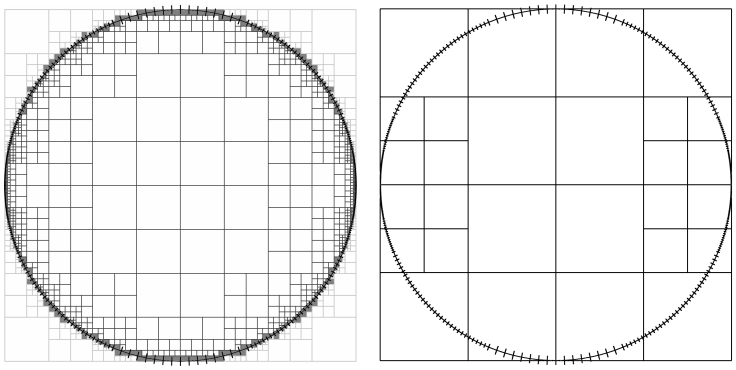

Figura 4: *Quadtree* de estimativa de carga (esquerda) e *quadtree* de decomposição (direita).

é construída de tal forma que a carga associada a cada uma de suas folhas seja menor do que uma carga máxima pré-definida. Esse limiar de carga é uma função da carga total e do número de processadores escravos disponíveis.

Se L representa a carga total, cada um dos N subdomínios deveria ter uma carga ideal de L/N. Isto acarretaria uma subdivisão regular da *quadtree* de decomposição em uma grade $\sqrt{N} \times \sqrt{N}$. É desejável que o número de subdomínios cobrindo a borda de entrada seja, pelo menos, igual ao número de processos escravos P. Esse número de subdomínios é calculado pela fórmula $4(\sqrt{N}-1)$, que representa o pior caso em que há $(\sqrt{N}-1)$ subdomínios por lado. Fazendo $P = 4(\sqrt{N}-1)$, descobre-se que a carga por subdomínio é de, no máximo, $L/(P/4+1)^2$.

A raiz da *quadtree* de decomposição é definida como um quadrado envolvendo o domínio, e sua carga é definida como a carga total estimada. Se a carga dessa célula for maior do que o limiar de carga, a célula é subdividida em quatro células de mesmo tamanho.

A carga associada a uma dada célula da *quadtree* de decomposição é estimada como o número de células-folha da *quadtree* de estimativa de carga que não estiverem fora do domínio e que estiverem dentro da célula da *quadtree* de decomposição. A verificação da carga máxima é feita recursivamente, para cada célula-folha da *quadtree* de decomposição e, sempre que a carga de uma célula exceder o limiar de carga, a célula é subdividida em quatro (direita da Figura 4). Cada folha da *quadtree* de decomposição que cruzar a fronteira ou que esteja dentro do domínio é considerada um subdomínio.

Os segmentos da fronteira de entrada são partilhados entre os subdomínios existentes. Logo, se os dois vértices de um segmento estiverem dentro do subdomínio, o segmento é dito pertencer exclusivamente a este subdomínio. Entretanto, se o segmento cruza dois ou mais subdomínios, ou toca a borda de alguns subdomínios, ele é dito pertencer a todos esses subdomínios. Não é necessário fazer nenhum tratamento especial nos segmentos das fraturas pois, para efeito de partilha entre os subdomínios, eles se comportam como segmentos normais da borda.

A utilização de uma *quadtree* para a decomposição do domínio e de uma *quadtree* diferente para a estimativa da carga é importante por três razões. Primeiro, as duas *quadtrees* não têm nenhum vínculo entre si, isto é, é possível usar qualquer método de estimativa de carga juntamente com a *quadtree* de decomposição de domínio, ou usar a *quadtree* de estimativa de carga combinada com qualquer técnica de decomposição do domínio. Na Seção 4, mostra-se que o uso das duas *quadtrees* juntas leva a decomposições do domínio nas quais as cargas de geração de malha dos subdomínios são bem estimadas. Segundo, já que a *quadtree* de estimativa de carga é geralmente muito refinada, não seria uma boa *quadtree* de decomposição. Se ela fosse utilizada para decompor o domínio, a carga associada a cada subdomínio seria muito pequena, e no, melhor dos casos, um *speed-up* muito pequeno seria observado. Terceiro, no procedimento

de deslocamento, a carga em cada subdomínio deve ser reestimada e, portanto, a *quadtree* de estimativa de carga deve ser modificada; entretanto, essa modificação não faz sentido no contexto da decomposição do domínio.

3.3 Balanceamento de Carga

Neste trabalho, os subdomínios são enviados aos processos escravos sob demanda. Inicialmente, o processo mestre retém todos os subdomínios. Um processo escravo requisita ao processo mestre um subdomínio e, assim que essa requisição é atendida, o processo escravo começa seu trabalho de geração de malha, enquanto que o processo mestre espera por outra requisição. Quando um processo escravo termina de trabalhar em um subdomínio, ele envia ao mestre outra requisição. Uma vez que todos os subdomínios tenham sido enviados aos processos escravos, o processo mestre espera pelos resultados.

3.4 Geração de Malha por um Processo Escravo

Uma vez determinados os subdomínios de um processador escravo, cada processo deve gerar uma malha para o seu subdomínio (esquerda da Figura 8), que é composto de uma fronteira e de uma caixa delimitadora (*bounding box*). As malhas locais (submalhas) nesses subdomínios podem ser geradas independemente umas das outras, usando, em cada processo, a AFT desenvolvida em Miranda et al. (1999, 2009); Cavalcante-Neto et al. (2001) por alguns dos co-autores deste trabalho. Estruturas de dados geométricas foram utilizadas para acelerar a busca por vértices candidatos para a geração de um novo triângulo e para a busca de possíveis arestas intersectantes, garantindo uma rápida execução do procedimento de geração de malha.

Com o objetivo de garantir que nenhum triângulo seja gerado fora dos limites do subdomínio, as seguintes condições foram adicionadas à AFT sequencial:

1. Um segmento que cruza a caixa delimitadora do subdomínio não deve avançar (esquerda da Figura 5).
2. Um segmento que não estiver estritamente dentro da caixa delimitadora do subdomínio não deve avançar se qualquer triângulo válido formado com ele cruzar a caixa delimitadora. Em outras palavras, considere a região de busca para a localização de um novo vértice que, juntamente com o segmento dado, forme um triângulo válido para a malha. Se esta região de busca cruzar a caixa delimitadora, então o segmento não deve avançar (direita da Figura 5).

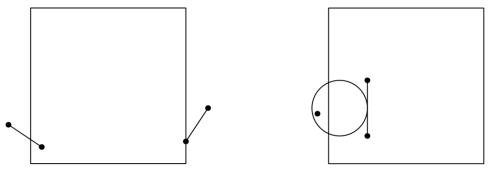

Figura 5: Modificações no avanço de fronteira: segmentos que cruzam a caixa delimitadora (esquerda) e região de busca que cruza a caixa delimitadora (direita).

As fraturas obedecem a essas mesmas condições. Assim, não é necessário nenhum tratamento especial. A Figura 6 denota as duas condições, no caso das fraturas. Os segmentos mais à esquerda não podem avançar, porque cruzam a caixa delimitadora do subdomínio, bem como os segmentos voltados para baixo, cuja região de busca cruza a caixa delimitadora.

Entretanto, por conta da existência de vértices com as mesmas coordenadas, o algoritmo de avanço de fronteira deve saber distinguir entre eles e escolher o vértice apropriado para a

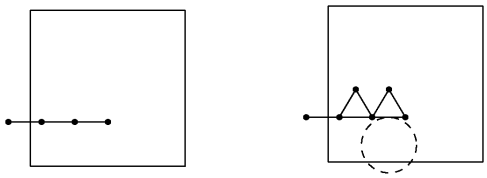

Figura 6: Fratura (esquerda) e condições adicionadas à AFT (direita).

formação de um novo triângulo (Figura 7). Essa distinção é feita utilizando-se as arestas da borda adjacentes a esses vértices. Quando dois vértices candidatos da fratura são selecionados para formar um novo elemento, aquele que estiver do mesmo lado da aresta base, com respeito à fratura, é escolhido. As normais das arestas adjacentes aos vértices são utilizadas para se fazer esse teste.

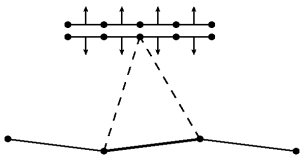

Figura 7: Aresta base (em destaque) avançando para um vértice de uma fratura.

Depois de avançar a fronteira recebida (esquerda da Figura 8) tanto quanto possível, os segmentos remanescentes compõem a nova fronteira (direita da Figura 8). Nesse processo, uma submalha é gerada (esquerda da Figura 9), sobre a qual uma melhoria pode ser aplicada (direita da Figura 9). Note que a nova fronteira não pode ser modificada quando a suavização ou qualquer outra técnica de otimização for realizada, por que não existe informação de adjacência suficiente para seus vértices ou segmentos.

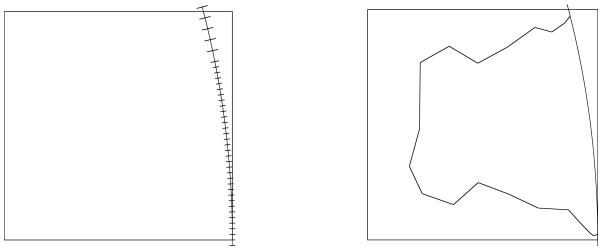

Figura 8: Fronteira e caixa delimitadora de um subdomínio: fronteira recebida (esquerda) e nova fronteira (direita).

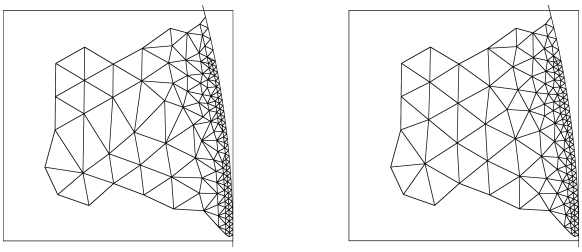

Figura 9: Malha de um subdomínio: malha sem melhorias (esquerda) e malha com melhorias (direita).

A fronteira recebida, exceto pelos segmentos que não avançaram, juntamente com a nova fronteira, formam um ou diversos polígonos (direita da Figura 8). Esses polígonos são utilizados

pelo processo escravo para classificar as células da *quadtree* de estimativa de carga. Logo, as células que estiverem estritamente dentro dos polígonos são consideradas fora do novo domínio, para o qual uma nova *quadtree* será futuramente gerada; e as células que cruzam a fronteira recebida também são consideradas fora do novo domínio. Na Figura 10, as células com arestas escuras estão dentro do novo domínio, as células preenchidas estão na borda, e as células com arestas claras estão fora do novo domínio.

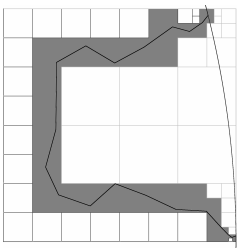

Figura 10: Nova classificação das células da *quadtree* de estimativa de carga.

Essa classificação será utilizada pelo processo mestre no próximo passo da técnica, juntamente com a nova fronteira. A malha gerada é mantida na memória do processo escravo.

3.5 Procedimento de Deslocamento

Assim que o processo mestre recebe a classificação e a nova fronteira de todos os subdomínios, ele atualiza a fronteira, inserindo os segmentos que apareceram recentemente e removendo os segmentos que foram avançados. O processo mestre também atualiza a classificação das células da *quadtree* de estimativa de carga. Visto que a *quadtree* de estimativa de carga não estava completamente refinada antes, como afirmado na Seção 3.1, ela tem que ser refinada agora para manter coerência com a fronteira. A parte da *quadtree* que representa a região estritamente fora da fronteira pode ser suprimida para se obter uma representação menos refinada (Figura 10).

Depois disso, cada célula-folha da *quadtree* de decomposição é deslocada de metade de seu tamanho em uma direção cartesiana, por exemplo, a direção X positiva (+X) (Figura 11).

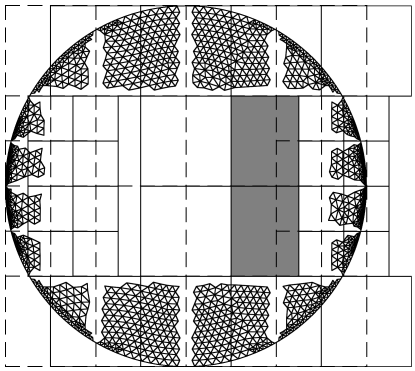

Figura 11: Submalhas geradas e células da *quadtree* de decomposição deslocadas na direção +X.

Com esse procedimento de deslocamento, é possível gerar mais malha entre dois subdomínios e avançar a fronteira para um subdomínio vizinho, o que não era possível quando se considerava a *quadtree* de decomposição não deslocada. O deslocamento de um subdomínio é realizado de forma que sua nova localização não seja distante demais de sua posição original, e ainda assim tenha uma quantidade significante de trabalho na geração de uma nova malha.

Note que duas células de decomposição deslocadas não devem se sobrepor, caso contrário, elementos intersectantes seriam gerados em subdomínios diferentes. Logo, se uma célula tem duas ou mais células vizinhas em uma direção de deslocamento, esta célula é distorcida em um formato retangular, de forma que a restrição da não-sobreposição seja satisfeita (células escuras e preenchidas da Figura 11). A aresta do subdomínio compartilhada por essas células vizinhas sofre o mesmo deslocamento que a menor célula vizinha, ou seja, aquela com maior profundidade na estrutura da *quadtree* de decomposição.

Depois de deslocar as células da *quadtree* de decomposição, o processo mestre partilha os segmentos da fronteira entre elas, recalcula suas cargas baseando-se na classificação da *quadtree* de estimativa de carga, e as envia aos processadores escravos para gerarem mais malha (Seções 3.2, 3.3 e 3.4).

O deslocamento é realizado sequencialmente nas seguintes direções cartesianas: para a direita (+X), para cima (+Y), para a esquerda (-X) e para baixo (-Y). Depois disso, a *quadtree* de decomposição retorna à sua posição original, e o ciclo é reiniciado, se necessário.

Note que, se uma célula de decomposição é deslocada em uma direção que não tem célula vizinha, ela não vai gerar nenhuma malha. Mais ainda, se uma célula de decomposição deslocada em uma direção (+Y, por exemplo) gera malha, suas células vizinhas naquela direção não precisam ser futuramente deslocadas na direção oposta (-Y, por exemplo), por que malha entre elas já foi previamente gerada. Isto não é sempre verdade já que, quando uma célula pequena é deslocada na direção de uma célula maior, o deslocamento da célula maior na direção oposta ainda pode gerar malha. Essas verificações básicas podem economizar tempo de computação e também tempo de comunicação.

3.6 Finalização da Malha

Depois do ciclo de deslocamento, o procedimento paralelo de geração de malha termina. Entretanto, cavidades sem malha podem ainda existir no interior do domínio (esquerda da Figura 12). Quando esse for o caso, o processo mestre finaliza o procedimento de geração de malha, aplicando uma AFT padrão com o objetivo de preencher as cavidades com triângulos.

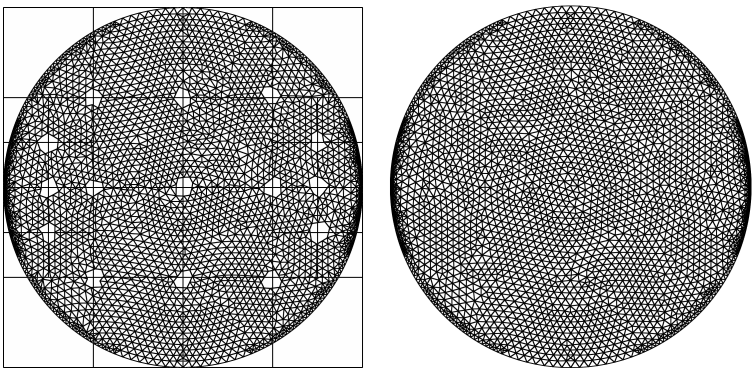

Figura 12: Cavidades remanescentes (esquerda) e malha final (direita).

Em seguida, o processo mestre aplica a mesma melhoria utilizada pelo gerador de malhas sequencial nas partes da malha que ainda não foram melhoradas pelos processos escravos. Essas partes consistem de segmentos da fronteira que passaram pelo processo mestre, juntamente com certas camadas de elementos adjacentes desses segmentos.

A camada 0 consiste dos próprios segmentos da fronteira, e a camada N compreende os elementos presentes na camada $N-1$ mais seus elementos adjacentes. Foi verificado (Ito et al.,

2007) que duas camadas de elementos são suficientes para uma boa malha. Depois desse passo, a malha está completamente gerada (direita da Figura 12).

4 EXEMPLOS

A técnica descrita neste trabalho foi implementada em C++, utilizando MPI para comunicação interprocessos. O computador utilizado para rodar os testes foi um *cluster* com 172 nós, cada um dos quais com dois processadores Intel® Core I7™ com quatro núcleos (3.0 GHz) e 24 GB de RAM.

4.1 Malhas

A Figura 13 mostra as bordas utilizadas pelo procedimento de geração em paralelo de malhas. Note que todos os exemplos apresentam fraturas. Subdomínios gerados com 8 processos são mostrados na Figura 14. Cada cor indica a parte da malha gerada por um processo diferente. As regiões finas coloridas localizadas entre dois subdomínios, como visto na Figura 14, ocorrem por conta do procedimento de deslocamento. Depois que dois subdomínios vizinhos são processados nos processos escravos, existe uma região estreita sem malha entre elas. Os elementos nesta região são gerados durante o ciclo de deslocamento. Se a tarefa de gerar malha para o subdomínio deslocado for atribuída a um dos processos que gerou malha em um dos subdomínios vizinhos, a cor da malha recentemente gerada é a mesma da malha gerada por um dos subdomínios vizinhos. Caso contrário, os elementos recebem uma cor diferente.

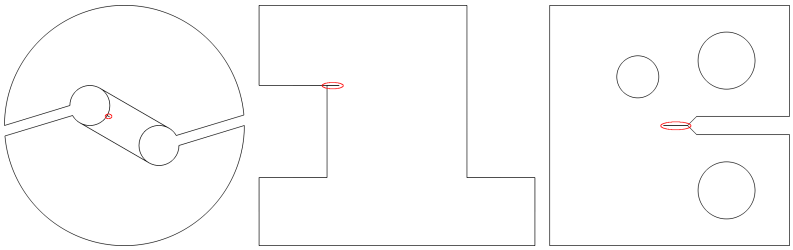

Figura 13: Bordas de entrada: Exemplo 1 (Spear et al., 2010) (esquerda), Exemplo 2 (Wawrzynek e Ingraffea, 1993) (centro) e Exemplo 3 (Miranda et al., 2003) (direita).

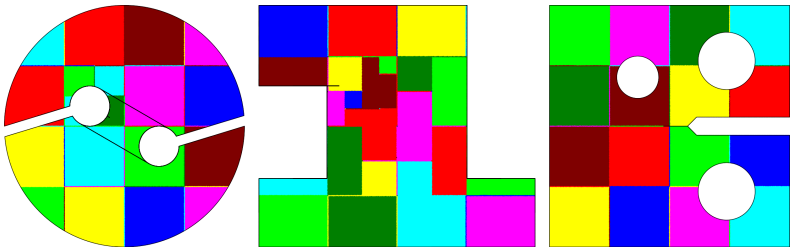

Figura 14: Subdomínios gerados com 8 processos: Exemplo 1 (esquerda), Exemplo 2 (centro) e Exemplo 3 (direita).

Note, nos Exemplos 1 e 2, que os subdomínios nas proximidades das fraturas são menores do que o restante dos subdomínios. Isso acontece porque elementos menores são desejáveis ao redor da fratura, levando a uma quantidade maior de elementos nessas áreas. Assim, uma carga maior é estimada, através de um maior refinamento da *quadtree* de estimativa de carga, implicando em subdomínios menores.

A Figura 15 destaca uma região entre dois subdomínios do modelo do Exemplo 2, com o objetivo de mostrar como as malhas geradas em paralelo se adequam perfeitamente. Como pode ser visto, as funções de tamanho e forma nos triângulos gerados foram respeitadas, mesmo que eles tenham sido gerados por diferentes processos em diferentes subdomínios, ou mesmo pelo processo mestre, no passo de finalização da malha (elementos em preto).

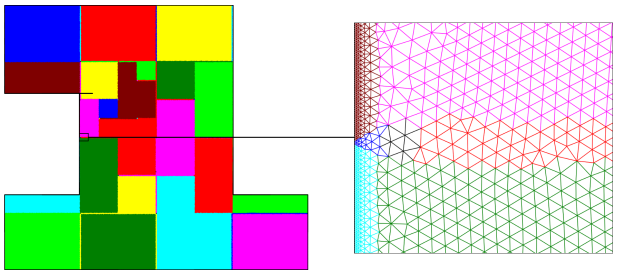

Figura 15: Região entre submalhas.

A Figura 16 destaca algumas áreas ao redor de fraturas. Como pode ser visto na esquerda Figura 16, apesar de a fratura estar partilhada por subdomínios diferentes, os elementos gerados respeitam as funções de tamanho e forma e têm boa qualidade. A direita da Figura 16 destaca uma fratura entre dois subdomínios. Neste caso, cada lado da fratura foi avançado por um subdomínio diferente. Perceba que isto não ocasiona nenhum conflito pois, apesar de compartilhados pelos dois subdomínios, as arestas que avançam em um subdomínio não avançam no outro.

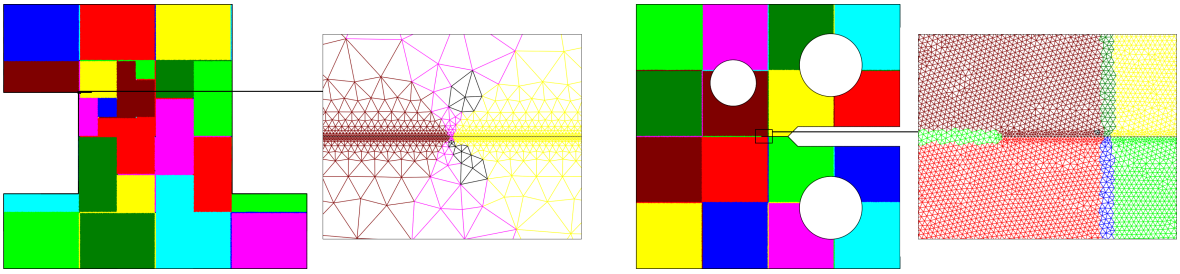

Figura 16: Subdomínios cruzando uma fratura (esquerda) e fratura compartilhada por diferentes subdomínios (direita).

A Figura 17 mostra os tamanhos das malhas geradas sequencialmente. Para a validação das ideias proposta neste trabalho, os tamanhos das malhas são razoáveis.

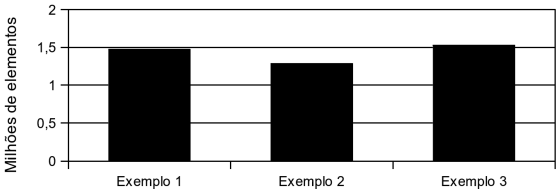

Figura 17: Tamanhos das malhas de exemplos.

4.2 Estimativa de Carga

Com o objetivo de provar que a estimativa de carga descrita na Seção 3.1 é precisa, o tempo de execução levado para gerar a malha em cada subdomínio foi calculado e comparado com a carga estimada para aquele subdomínio. A Figura 18 compara o tempo de execução de cada passo do procedimento de avanço de fronteira e o tempo de execução estimado para alguns subdomínios. Representar todos os subdomínios não caberia no gráfico, além de não ser necessário para esta comparação. Um subdomínio pode aparecer repetido, mas cada vez corresponde a uma posição de deslocamento diferente.

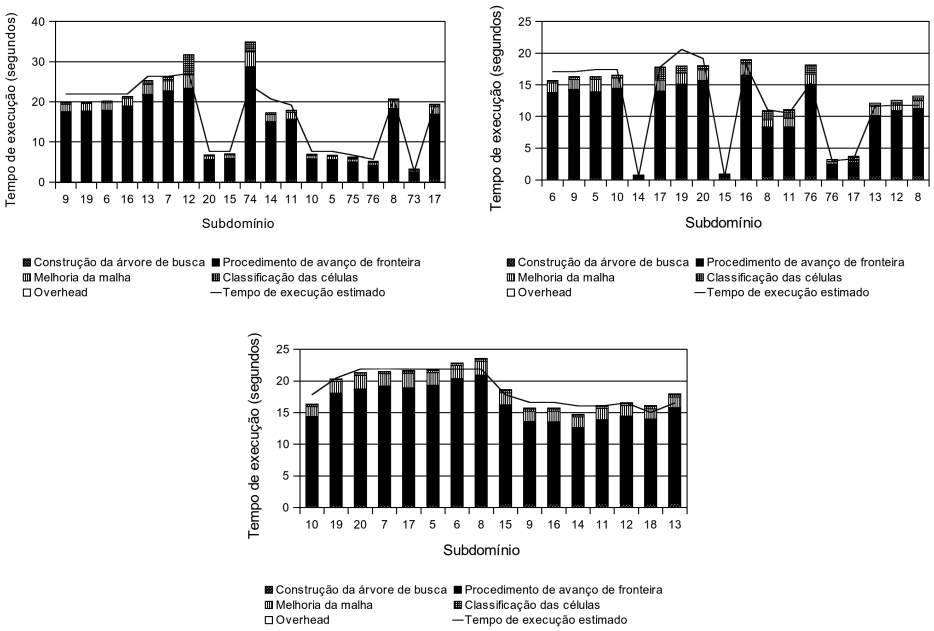

Figura 18: Estimativa de tempo de execução dos subdomínios: Exemplo 1 (superior esquerdo), Exemplo 2 (superior direito) e Exemplo 3 (inferior).

Os passos de Construção da árvore de busca, de Procedimento de avanço de fronteira e de Melhoria da malha pertencem ao procedimento de geração de malha empregado, e o passo de Classificação das células foi descrito na Seção 3.4. O passo de *Overhead* contabiliza o tempo extra gastado fazendo trabalho improdutivo.

As linhas na Figura 18 representam o tempo de execução estimado, que foi calculado como a carga escalonada pelo fator $\bar{t}/\bar{l}$, onde $\bar{t}$ é o tempo médio de execução e $\bar{l}$ é a carga média estimada para os subdomínios. Já que somente a proporção entre as cargas é necessária para o balanceamento de carga, este escalonamento fixo pode ser aplicado para propósitos de avaliação. A Figura 19 mostra as mesmas comparações, mas considerando o número de elementos e o número de vértices em vez do tempo de execução. Como pode ser visto, pode-se considerar que a estimativa é bem precisa.

A Figura 20 mostra o erro entre o tempo de execução estimado e o tempo de execução efetivo para cada subdomínio, em porcentagem. Um erro positivo significa que o tempo efetivo foi menor que o tempo estimado. Um erro negativo significa que o tempo efetivo foi maior do que o tempo estimado. O valor absoluto do erro, portanto, denota qual a diferença entre o tempo estimado e o tempo efetivo. O erro absoluto máximo mostra a pior estimativa de tempo. A figura também mostra a média e o desvio padrão para o erro absoluto.

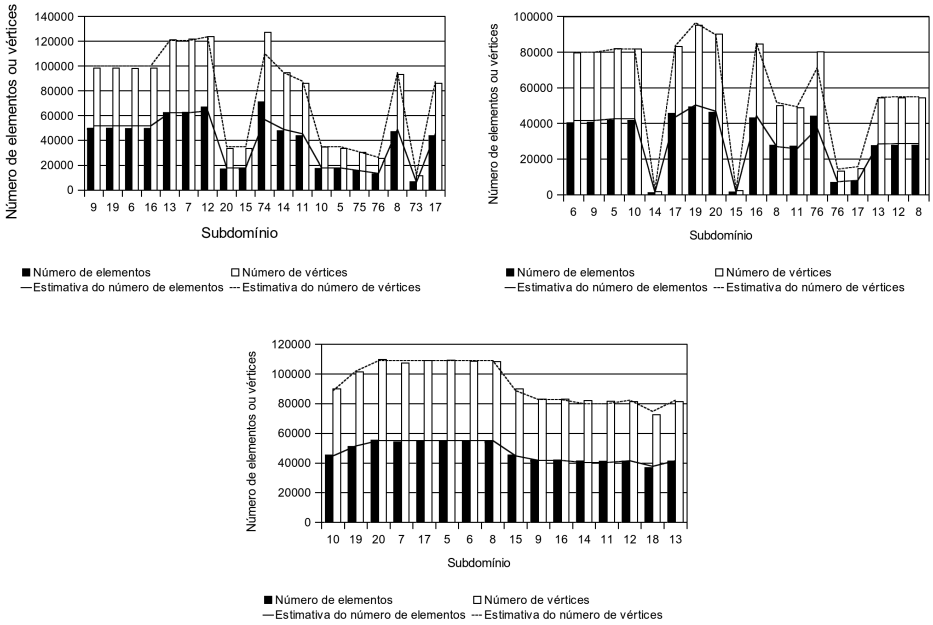

Figura 19: Estimativa do número de elementos e de vértices para os subdomínios: Exemplo 1 (superior esquerdo), Exemplo 2 (superior direito) e Exemplo 3 (inferior).

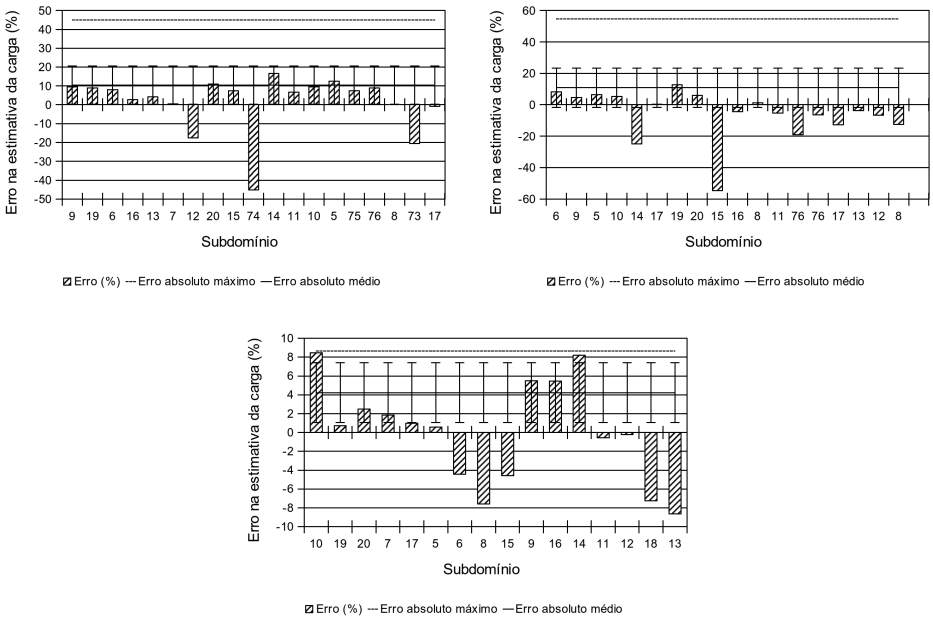

Figura 20: Erro entre a carga estimada e o tempo de execução efetivo para os subdomínios: Exemplo 1 (superior esquerdo), Exemplo 2 (superior direito) e Exemplo 3 (inferior).

A Tabela 1 apresenta os valores de média e de desvio padrão para a diferença entre o tempo de execução estimado e o efetivo ($\bar{T}$ e $\sigma(T)$). Essa mesma análise foi realizada para o número de elementos ($\bar{E}$ e $\sigma(E)$) e o número de vértices ($\bar{V}$ e $\sigma(V)$) gerados em um subdomínio. Esses valores mostram que o erro no tempo de execução é, em média, menor do que $17,0\%$. Para o número de elementos e de vértices, o erro é, em média, menor do que $7,4\%$.

Como indicado na Tabela 1, o método empregado, baseado em *quadtree*, produz uma esti-

Tabela 1: Estatísticas de erro na estimativa da carga.

Erro absoluto		Exemplo 1	Exemplo 2	Exemplo 3
Tempo	$\bar{T}$	$10,4\%$	$10,8\%$	$4,2\%$
	$\sigma(T)$	$10,1\%$	$12,5\%$	$3,2\%$
Elementos	$\bar{E}$	$4,5\%$	$5,4\%$	$1,0\%$
	$\sigma(E)$	$5,2\%$	$5,3\%$	$0,8\%$
Vértices	$\bar{V}$	$2,9\%$	$4,7\%$	$1,1\%$
	$\sigma(V)$	$3,5\%$	$9,0\%$	$0,9\%$

mativa mais precisa para o número de elementos do que para o tempo de execução. Isso deve-se ao fato de que a complexidade de um algoritmo de avanço de fronteira não é linear ($O(N)$) no número de elementos gerados, mas da ordem de $O(N \log N)$ (Löhner e Parikh, 1988), com N sendo o número de elementos gerados. Assim, essa maior imprecisão na estimativa do tempo de execução advém do termo $\log N$ presente na complexidade. Uma maior investigação sobre esse assunto deve ainda ser realizada. Entretanto, todos esses dados mostram que o método empregado é uma boa predição para a carga de um subdomínio.

4.3 Balanceamento de Carga

A Figura 21 mostra o desempenho de cada processador para os primeiros subdomínios recebidos, antes do primeiro deslocamento das células da *quadtree* de decomposição, com oito processadores. Como pode ser visto, a quantidade de trabalho está bem distribuída entre os processadores, indicando que a carga total está bem balanceada.

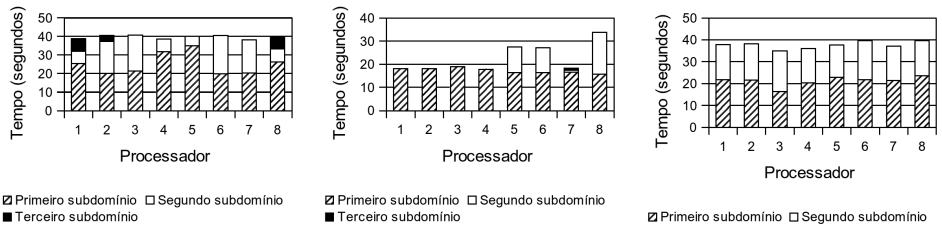

Figura 21: Desempenho de cada processador: Exemplo 1 (esquerdo), Exemplo 2 (centro) e Exemplo 3 (direito).

No segundo exemplo, devido à existência de 11 subdomínios com muita carga e apenas oito processadores, um desbalanceamento seria inevitável. Como se pode ver, o processador responsável pelo mais leve dos oito subdomínios mais pesados foi o oitavo processador. Esse processador foi ainda o responsável pelo mais pesado dos subdomínios restantes. Qualquer outra distribuição em que esses dois subdomínios ficassem em processadores diferentes aumentaria ainda mais o tempo de execução total, pois um deles ou algum subdomínio mais pesado acabaria sendo enviado para um processador que já estava responsável por outro subdomínio ainda mais pesado. Isto indica que o balanceamento de carga por demanda utilizado neste trabalho minimiza o desbalanceamento de carga.

4.4 *Speed-up*

Os gráficos da Figura 22 mostram o tempo de execução juntamente com o *speed-up* atingido pela implementação da técnica. O *speed-up* é uma métrica que diz o quão mais rápida uma implementação paralela é quando comparada com sua versão sequencial, e é calculado como

T_s/T_n, onde T_s é o tempo que a implementação sequencial levou para acabar e T_n é o tempo que a implementação paralela levou para acabar quando foi executado com n processadores. Idealmente, uma implementação paralela teria um *speed-up* linear, significando que n processadores a tornam n vezes mais rápida. Na prática, isso é difícil de ser obtido, devido a inevitáveis porções sequenciais presentes no algoritmo paralelo.

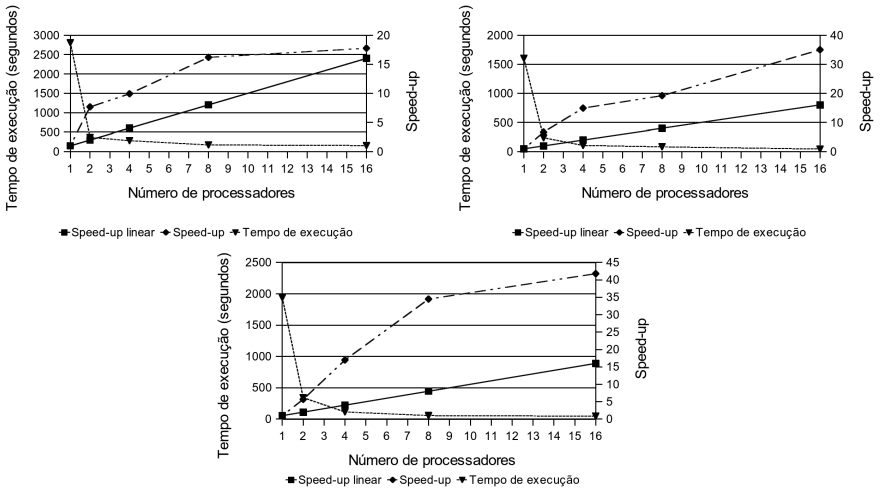

Figura 22: Tempo de execução e *speed-up*: Exemplo 1 (superior esquerdo), Exemplo 2 (superior direito) e Exemplo 3 (inferior).

Como visto na Figura 22, um *speed-up* superlinear foi frequentemente obtido pelo algoritmo. Isso acontece porque, para cada subdomínio, a fronteira, a malha, e a área a ser preenchida por triângulos são consideravelmente pequenos, quando comparados com a entrada completa, utilizada pela implementação sequencial. Portanto, as operações de inserção e remoção de segmentos da fronteira, a melhoria da malha e as buscas geométricas acontecem muito mais rápido, quando comparados com a implementação sequencial. Como também pode ser visto na mesma figura, os tempos de execução têm um decaimento excepcional.

4.5 Qualidade

Para certificar que a malha gerada pela técnica descrita tem uma boa qualidade, a métrica utilizada foi $\alpha = 2R_i/R_c$, onde R_i e R_c são os raios dos círculos inscrito e circunscrito a um triângulo da malha, respectivamente. Essa métrica α tem valor $1,0$ para um triângulo equilátero. Quanto pior for a qualidade de um elemento, mais próximo de $0,0$ é o valor de α. Um elemento com $\alpha \leq 0,1$ é dito ter uma qualidade muito pobre, enquanto que elementos com $\alpha \geq 0,7$ têm boa qualidade.

Os gráficos da Figura 23 mostram a qualidade das malhas geradas pelas diversas execuções do algoritmo. Devido ao não-determinismo inerente de algoritmos paralelos, diferentes execuções de um dado programa com a mesma entrada e os mesmos parâmetros podem gerar saídas diferentes. Portanto, é necessário analisar a qualidade de todas as malhas geradas.

Como pode ser visto na Figura 23, em todos os casos, as malhas geradas apresentam boa qualidade. Somente uma pequena porcentagem de elementos apresentam qualidade pobre. A Figura 24 indica o quanto as malhas geradas em paralelo desviam-se da malha gerada sequencialmente. Para o Exemplo 1, no máximo $0,24\%$ dos elementos têm valores diferentes para α, comparados com a malha sequencial. As malhas do Exemplo 2 e do Exemplo 3 apresentaram desvios de $0,48\%$ e $0,14\%$, respectivamente.

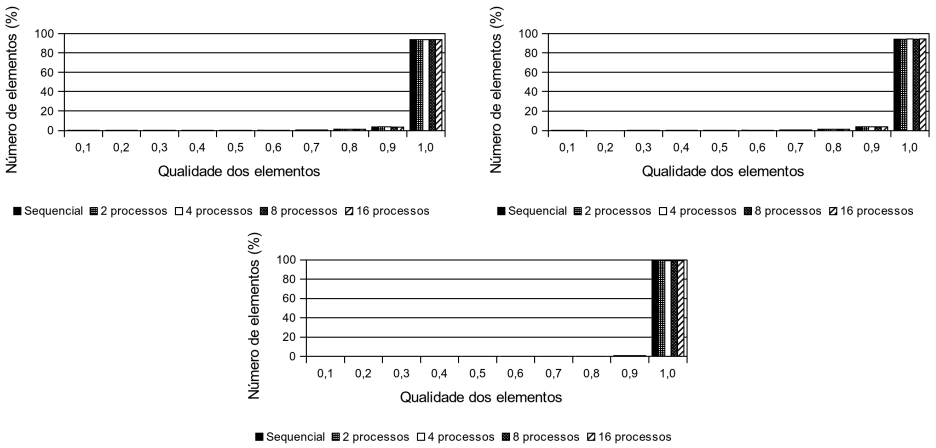

Figura 23: Qualidade das malhas: Exemplo 1 (superior esquerdo), Exemplo 2 (superior direito) e Exemplo 3 (inferior).

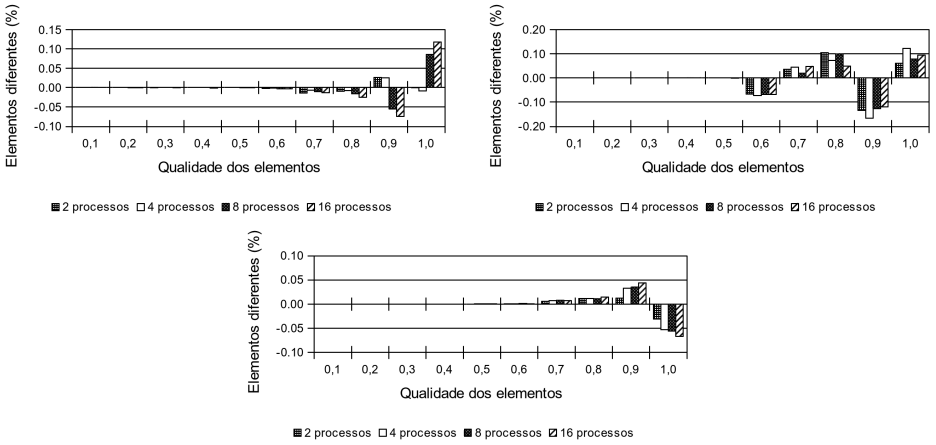

Figura 24: Diferença do número de elementos em cada intervalo de qualidade, em porcentagem: Exemplo 1 (superior esquerdo), Exemplo 2 (superior direito) e Exemplo 3 (inferior).

Isto mostra que a técnica paralela descrita neste trabalho gera uma malha de qualidade aproximadamente igual à da malha gerada sequencialmente, que tem uma qualidade muito boa devido às melhorias de qualidade executadas sequencialmente. Note também que um número maior de subdomínios geralmente leva a uma diferença maior, apesar de esta diferença não chegar a $0,5\%$ dos elementos da malha.

5 CONCLUSÕES

Este trabalho apresenta uma técnica para gerar malhas em paralelo utilizando o modelo mestre/escravos. A estimativa da carga é dada por uma *quadtree* construída para a borda dada como entrada. O domínio é decomposto em diversos subdomínios quadrados, também baseados em uma *quadtree*, que leva em conta a carga e o número de processos escravos do sistema paralelo. Esses subdomínios são enviados para os processos escravos, que geram malhas empregando um algoritmo sequencial de geração de malhas.

Depois disso, a fronteira resultante é enviada de volta para o processo mestre, que desloca as células da *quadtree* de decomposição em uma direção cartesiana, reparticiona a fronteira, e envia os subdomínios em suas novas posições para os processos escravos que, mais uma

vez, geram malhas sequencialmente. Esse procedimento de deslocamento/geração de malha continua até que não seja possível gerar-se mais malha, cada vez deslocando as células da *quadtree* de decomposição em direções diferentes.

Essa técnica apresenta uma boa estimativa de carga, e um bom balanceamento da carga. A técnica paralela mostrou um bom *speed-up*, mais que linear, e as malhas geradas em paralelo têm a mesma qualidade das geradas sequencialmente.

Alguns pontos ainda são de interesse nesta pesquisa. A quantidade de trabalho sequencial realizada no processo de finalização da malha poderia ser reduzida empregando deslocamentos diferentes, livrando-se das cavidades tais como as mostradas na esquerda da Figura 12; e aplicando a melhoria das camadas de elementos em paralelo, aproveitando melhor os deslocamentos cartesianos.

Também é possível remover a sincronização global presente no deslocamento da *quadtree* de decomposição. Para deslocar uma célula em uma direção, é necessário que uma partição conheça somente os seus vizinhos naquela direção, juntamente com seus segmentos de fronteira. Portanto, um deslocamento local pode ser empregado somente entre os subdomínios envolvidos.

A ideia apresentada neste trabalho pode também ser naturalmente estendida para a geração de malhas tridimensionais, e os autores estão atualmente trabalhando em uma versão 3D desta técnica.

Agradecimentos

O primeiro autor agradece à CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pela bolsa de estudos concedida. O terceiro autor agradece o suporte do CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) através da bolsa de produtividade em pesquisa 305596/2010-1. Agradecemos à ANP, à Petrobras e ao Laboratório de Computação Científica e Visualização da Universidade Federal de Alagoas por garantir acesso aos recursos computacionais do *cluster* GradeBR/UFAL da Rede Galileu.

REFERÊNCIAS

Cavalcante-Neto J.B., Wawrzynek P.A., Carvalho M.T.M., Martha L.F., e Ingraffea A.R. An algorithm for three-dimensional mesh generation for arbitrary regions with cracks. *Engineering with Computers*, 17(1):75–91, 2001. doi:10.1007/PL00007196.

Chernikov A. e Chrisochoides N. Parallel 2D graded guaranteed quality Delaunay mesh refinement. In *Proceedings of the 14th International Meshing Roundtable*. Sandia National Laboratory, 2005.

Chernikov A.N. e Chrisochoides N.P. Parallel guaranteed quality Delaunay uniform mesh refinement. *SIAM Journal on Scientific Computing*, 28(5):1907–1926, 2006. doi:10.1137/050625886.

Chrisochoides N. Parallel mesh generation. *Numerical Solution of Partial Differential Equations on Parallel Computers*, 51:237–259, 2005.

Chrisochoides N. e Nave D. Simultaneous mesh generation and partitioning for Delaunay meshes. *Mathematics and Computers in Simulation*, 54(4-5):321–339, 2000.

De Cougny H.L. e Shephard M.S. Parallel volume meshing using face removals and hierarchical repartitioning. *Computer Methods in Applied Mechanics and Engineering*, 174(3-4):275–298, 1999. doi:10.1016/S0045-7825(98)00300-4.

Hodgson D.C. e Jimack P.K. Efficient parallel generation of partitioned, unstructured meshes. *Advances in Engineering Software*, 27(1-2):59–70, 1996. doi:10.1016/0965-9978(96)

00008-7.

Ito Y., Shih A.M., Erukala A.K., Soni B.K., Chernikov A., Chrisochoides N.P., e Nakahashi K. Parallel unstructured mesh generation by an advancing front method. *Mathematics and Computers in Simulation*, 75(5-6):200–209, 2007. doi:10.1016/j.matcom.2006.12.008.

Karypis G. e Kumar V. *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Version 4.0*. University of Minnesota, 1998.

Karypis G., Schloegel K., e Kumar V. *ParMETIS: Parallel Graph Partitioning and Sparse Matrix Ordering Library, Version 3.2*. University of Minnesota, 2011.

Kohout J., Kolingerová I., e Žára J. Parallel Delaunay triangulation in E^2 and E^3 for computers with shared memory. *Parallel Computing*, 31(5):491–522, 2005. doi:10.1016/j.parco.2005.02.010.

Lämmer L. e Burghardt M. Parallel generation of triangular and quadrilateral meshes. *Advances in Engineering Software*, 31(12):929–936, 2000. doi:10.1016/S0965-9978(00)00068-5.

Löhner R. A parallel advancing front grid generation scheme. *International Journal for Numerical Methods in Engineering*, 51(6):663–678, 2001. doi:10.1002/nme.175.

Löhner R. e Parikh P. Generation of three-dimensional unstructured grids by the advancing-front method. *International Journal for Numerical Methods in Fluids*, 8:1135–1149, 1988.

Miranda A.C.O., Cavalcante-Neto J.B., e Martha L.F. An algorithm for two-dimensional mesh generation for arbitrary regions with cracks. In *SIBGRAPI '99: Proceedings of the XII Brazilian Symposium on Computer Graphics and Image Processing*, páginas 29–38. IEEE Computer Society, 1999.

Miranda A.C.O., Martha L.F., Wawrzynek P.A., e Ingraffea A.R. Surface mesh regeneration considering curvatures. *Engineering with Computers*, 25(2):207–219, 2009. ISSN 0177-0667. doi:10.1007/s00366-008-0119-9.

Miranda A.C.O., Meggiolaro M.A., Castro J.T.P., Martha L.F., e Bittencourt T.N. Fatigue life and crack path predictions in generic 2D structural components. *Engineering Fracture Mechanics*, 70:1259–1279, 2003.

Okusanya T. e Peraire J. Parallel unstructured mesh generation. In *Proceedings of the 5th International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, páginas 719–729. Mississippi State University, 1996.

Rivara M.C., Calderon C., Fedorov A., e Chrisochoides N. Parallel decoupled terminal-edge bisection method for 3D mesh generation. *Engineering with Computers*, 22:111–119, 2006. doi:10.1007/s00366-006-0013-2.

Spear A.D., Veilleux M.G., Bozek J.E., e Ingraffea A.R. Structural assessment and prognosis using a multi-scale, fracture mechanics-based approach. In *Proceedings of the 2010 Aircraft Airworthiness and Sustainment Conference*. Texas, United States, 2010.

Wawrzynek P.A. e Ingraffea A.R. *FRANC2D - A Two-Dimensional Crack Propagation Simulator - User's guide - Version 3.1*, 1993.

Wilson J.K. e Topping B.H.V. Parallel adaptive tetrahedral mesh generation by the advancing front technique. *Computers & Structures*, 68(1-3):57–78, 1998. doi:10.1016/S0045-7949(98)00042-X.

Zagaris G., Pirzadeh S.Z., e Chrisochoides N. A framework for parallel unstructured grid generation for practical aerodynamic simulations. In *Proceedings of the 47th AIAA Aerospace Sciences Meeting*. AIAA - American Institute of Aeronautics and Astronautics, 2009.