4 Algoritmo de Multi-resolução Proposto para RNPs

Este capítulo tem por objetivo propor um modelo de multi-resolução para RNPs, levando em consideração as características geométricas das malhas de simulação, a vizinhança IJK entre células, as propriedades escalares associadas a cada célula do reservatório e as características de visualização discutidas anteriormente.

O MMR é proposto na Seção 4.1, a Seção 4.2 detalha a operação local utilizada pelo modelo e a Seção 4.3 faz algumas considerações a respeito da estrutura de MR utilizada.

4.1 Hierarquia de Células Hexaédricas

Visando atender adequadamente às características de visualização descritas na Seção 3.4, o modelo de MR adotado é hierárquico, possuindo as características de indexação espacial e agrupamento com coerência espacial encontradas na maioria dos MMRHs, em particular, em [Gar98]. A célula hexaédrica foi utilizada como base do modelo proposto devido à sua importante semântica.

Para permitir a extração de malhas adaptativas sem causar a diminuição do poder de expressão do MMRH, são permitidas aberturas e pequenas interseções entre as células nas aproximações geradas. Isso pode alterar a topologia da malha de simulação

original, mas não é um fato crítico devido à existência típica de degraus e falhas nas malhas.

Apesar do algoritmo proposto se basear na vizinhança IJK, ele não segue uma hierarquia pré-definida, a exemplo da *octree*. As células são colapsadas em função do custo (erro) do colapso.

Inspirado nas técnicas de baixo para cima (*bottom-up*) incrementais de simplificação, normalmente relacionadas a MMRRs, o algoritmo de construção da estrutura hierárquica de multi-resolução aqui proposto consiste na escolha de uma série de operações incrementais que formam um histórico de operações, o qual, a exemplo das malhas progressivas [Hop96], leva do modelo mais refinado do reservatório até o seu modelo menos refinado. A partir deste histórico é possível identificar as dependências cronológicas entre operações e codificá-las através de uma árvore.

4.1.1 Colapso de Células

A operação utilizada pelo algoritmo proposto neste trabalho é uma operação de simplificação. Ela recebe duas células, $\boldsymbol{a}$ e $\boldsymbol{b}$, que abrangem a região do espaço ($\boldsymbol{a} \cup \boldsymbol{b}$), e retorna uma nova célula, $\boldsymbol{c}$, e o erro, $\boldsymbol{\varepsilon}$, associado à substituição de $\boldsymbol{a}$ e $\boldsymbol{b}$ por $\boldsymbol{c}$ (Figura 4.1). A esta aproximação no espaço geométrico está relacionada a propriedade no espaço paramétrico:

$$\boldsymbol{dom}(\boldsymbol{c}) = \boldsymbol{dom}(\boldsymbol{a} \cup \boldsymbol{b}) \text{ ,} \qquad \textbf{P.4.1}$$

A operação é denominada **colapso de células**, ou simplesmente **colapso**, e as células geradas por ele são denominadas **macro-células**.

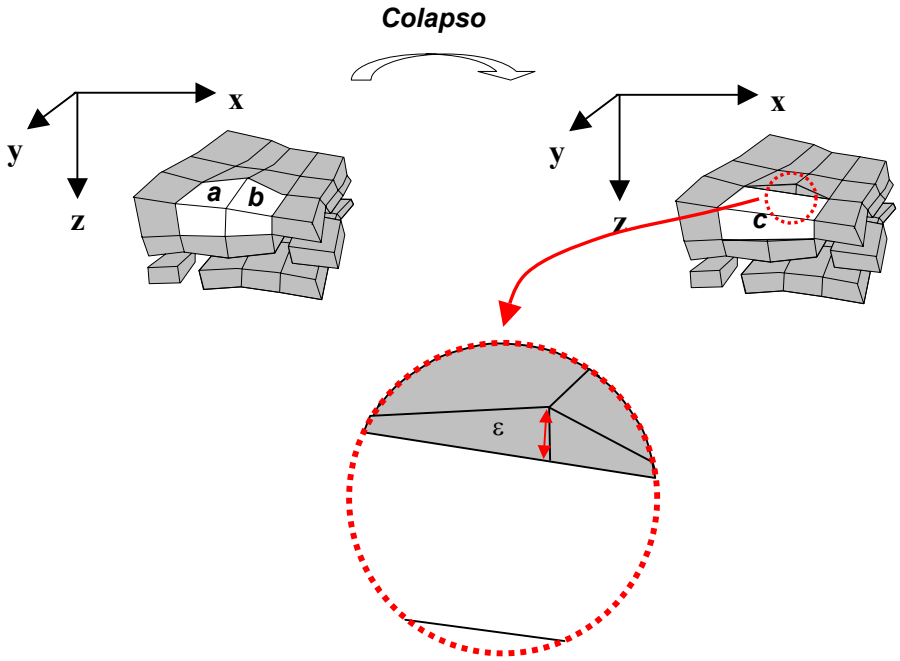

Figura 4.1 - Operação incremental de colapso. O colapso calcula a geometria da macro-célula, à direita, a partir das duas células, à esquerda. O erro ε, que nesta ilustração é o próprio erro geométrico, é associado ao colapso.

A Figura 4.2 mostra um exemplo bidimensional de abertura oriundo de uma operação de colapso.

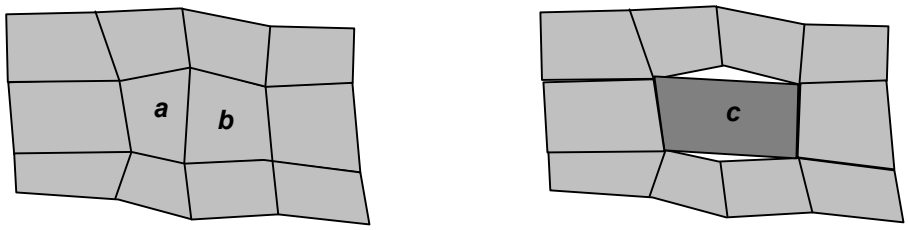

Figura 4.2 - Versão bidimensional da operação de colapso. A operação altera a borda de *(a ∪ b)* e, portanto, como neste exemplo, pode modificar a topologia fora do domínio ao qual é aplicada.

4.1.2 Algoritmo Guloso para Construção do Espaço de Modelos

O algoritmo proposto para a construção do espaço de modelos gerencia a construção de um histórico cronológico de operações de colapso, que são efetuadas uma a uma, levando a diferentes aproximações do reservatório que variam localmente uma da outra (Figura 4.3). Para criá-lo é utilizado um algoritmo guloso que iterativamente seleciona o colapso de menor erro.

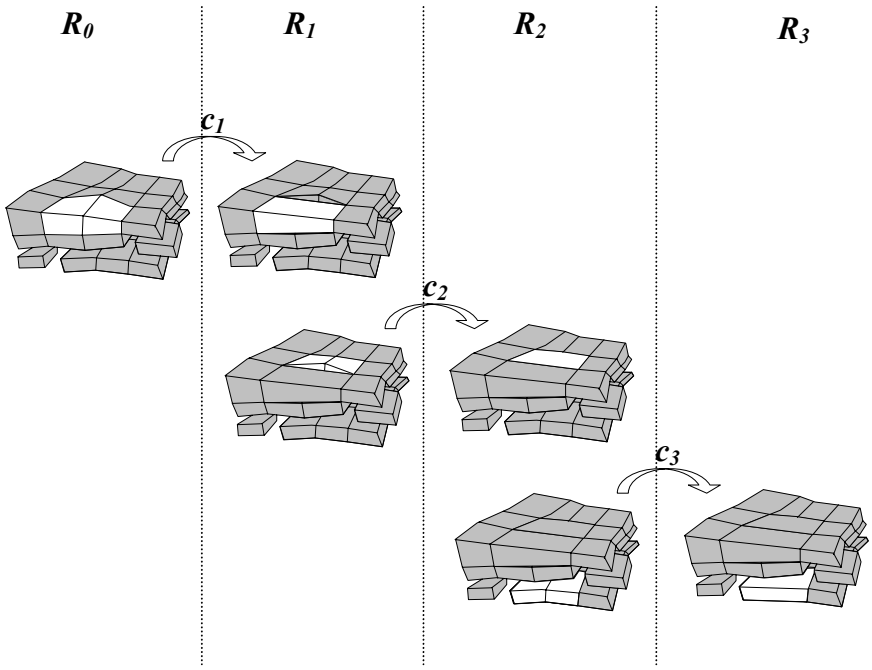

Figura 4.3 – Histórico. O histórico $\{c_1, c_2, c_3\}$ leva, a partir R_0, a mais três representações distintas do reservatório: R_1, R_2, R_3.

O algoritmo, em alto nível, pode ser especificado através da máquina de estado ilustrada na Figura 4.4. O estado inicial, E_0, é formado pelo reservatório original, R_0, e por um histórico vazio, ∅. Após a escolha do colapso de menor custo, a inserção deste no histórico e a efetuação do colapso, tem-se o próximo estado, E_1, que é formado por uma aproximação R_1 do reservatório original e por um histórico que, agora, contém um colapso a mais, $\{c_1\}$ (Figura 4.6). O algoritmo continua até que não existam mais colapsos a serem efetuados.

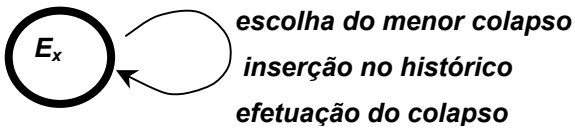

Figura 4.4 - Máquina de estado que representa o algoritmo guloso.

Dado um reservatório com n células, o número total de colapsos possíveis, sem considerar a vizinhança IJK, é n^2. Entretanto, considerando a vizinhança, que é explicitada pelo grafo de adjacência A, este número é próximo de $3n$, nunca excedendo-o. Uma vez que células vizinhas estão próximas no espaço geométrico, os colapsos entre elas possuem os menores erros, sendo provavelmente os mais relevantes para o algoritmo.

Assim, A pode ser interpretado de uma nova forma: seus nós continuam representando células, mas, agora, seus arcos representam colapsos e a cada arco está associado um custo que corresponde ao erro atribuído ao colapso.

Para extrair eficientemente de A o arco de menor custo, pode-se utilizar uma fila de prioridades, particularmente um *heap*, onde cada elemento é um arco. O *heap* pode ser criado em tempo linear no número de arcos a [Cor90]. Operações de extração do arco de menor custo e atualização de um elemento podem ser feitas em $O(\log a)$.

Devido à localidade da operação de colapso, a cada iteração do algoritmo, são necessárias apenas alterações locais nas estruturas de A e do *heap* para mantê-las consistentes com o próximo estado do algoritmo. A efetuação de um colapso faz com que o grafo de adjacência seja alterado e que um novo nó, que representa a macro-célula, seja criado, herdando a vizinhança dos nós colapsados (Figura 4.5). Assim, é necessário atualizar a vizinhança do novo nó recalculando custos. Isto pode ser feito em tempo proporcional ao grau g do novo nó. Além disso, é necessário tornar o *heap* consistente e atualizar os elementos cujos custos foram modificados. Isto pode ser feito em $O(g \log a)$.

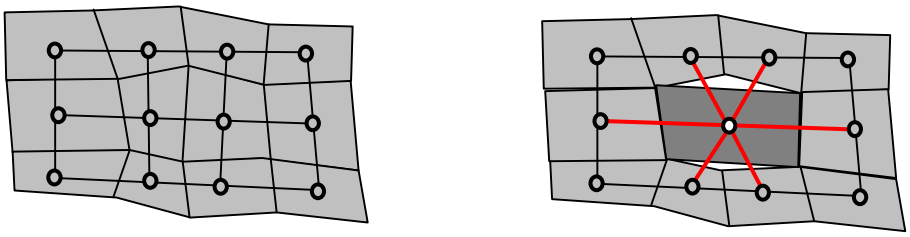

Figura 4.5 – Vizinhança. O novo nó herda a vizinhança dos nós colapsados. O grau do novo nó é igual à soma dos graus dos nós colapsados menos os colapsados que se tornaram redundantes menos 1.

Neste momento, vale fazer uma observação relativa à propriedade INAT, que indica se uma célula é ativa ou inativa. Como foi visto no Capítulo 3, a geometria e os valores de propriedade associados a uma célula inativa devem ser desprezados, fato este que levaria à existência de colapsos inócuos. Isso é evitado pelo algoritmo através de uma modificação no grafo ***A***: se uma célula é vizinha de uma célula inativa, esta vizinhança é alterada de forma que a nova vizinha seja a próxima célula ativa naquela direção. Dessa forma, alguns colapsos são eliminados e o número de arcos ***a*** do grafo passa a ser próximo a 3***n'***, onde ***n'*** é o número de células ativas.

O pseudo-código do algoritmo é o seguinte:

1° passo: *constrói o grafo de adjacência eliminando células inativas //O(**n**)*
2° passo: *constrói o heap //O(**a**)*
3° passo: *constrói o histórico vazio //O(1)*
4° passo: *enquanto houver colapsos*
{
    *extrai o menor colapso //O(log**a**)*
    *atualiza o grafo de vizinhança //O(**g**)*
    *atualiza o heap //O(**g**log**a**)*
    insere no histórico //O(1)
}

Como ***n*** difere de ***n'*** pela soma de uma constante e, uma vez que ***a*** é proporcional a ***n'***, a complexidade[9] final do algoritmo é O(***n'*** log ***n'***), desde que a variável ***g*** seja limitada superiormente por uma constante. Para isso, o algoritmo controla o grau dos

nós do grafo de adjacência durante a iteração. Assim, todo colapso que geraria um novo nó com grau acima de um valor limite ***G*** é proibido de ser efetuado, recebendo um custo elevado.

Cada colapso reduz em uma unidade o número de células. Logo, são efetuados ***n'***-1 colapsos sendo criadas exatamente ***n'***-1 macro-células. Isto implica em um histórico de complexidade O(***n'***).

---

[9] *A prova completa da complexidade deste tipo de algoritmo guloso pode ser encontrada em [Garland99].*

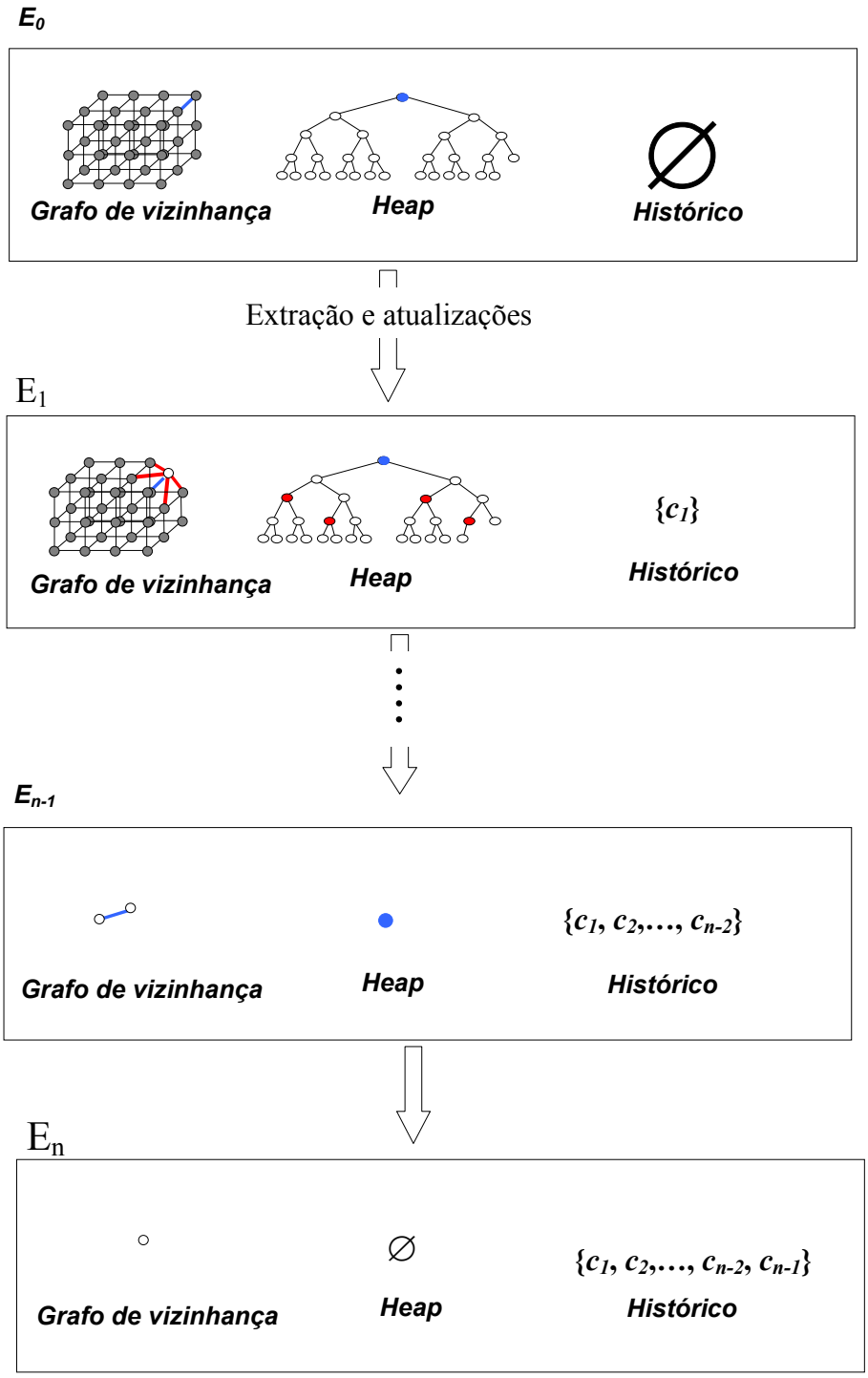

Figura 4.6 - Algoritmo incremental. O algoritmo inicia a partir de E_0 e, a cada passo, extrai do *heap* e efetua o colapso de menor custo (em azul) criando uma nova célula (em branco) no grafo de adjacência. Além disso, atualiza a estrela do novo nó (em vermelho).

4.1.3 Hierarquia

O histórico gerado pelo algoritmo incremental apenas codifica de forma seqüencial as operações de colapso, criando dependências cronológicas entre elas. Entretanto, através da análise de interferência das operações entre si, podem-se identificar operações independentes passíveis de serem efetuadas simultaneamente.

A exemplo da hierarquia de vértices [Hop97, Lue97, Xia96], pode-se construir em tempo O(***n'***), diretamente a partir do histórico de operações, uma estrutura que codifica a série de colapsos e suas dependências. A estrutura resultante é uma árvore binária (Figura 4.7) que, assim como a hierarquia de vértices, é apropriada para a extração de modelos adaptativos em tempo real. Cada nó da árvore corresponde a uma célula e dois arcos irmãos a uma operação de colapso.

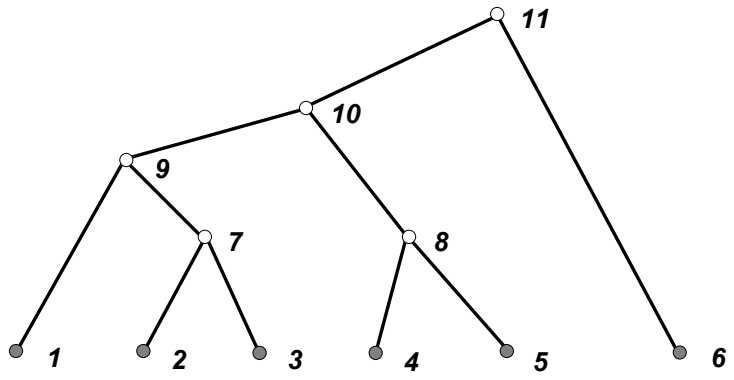

Figura 4.7 - Estrutura da hierarquia de células. As folhas correspondem às células do modelo original. Os nós internos correspondem às macro-células.

Na verdade, a hierarquia pode ser construída durante a execução do algoritmo incremental sem a necessidade de existir explicitamente um histórico. Para isto, é necessário gerenciar uma **floresta** de hierarquias de células. A floresta inicial é formada por tantas árvores quantas forem as células ativas do reservatório. Cada árvore, nesta etapa inicial, é formada por um único nó. Durante o laço de repetição, quando um colapso é efetuado, duas árvores são fundidas. Ao final do laço de repetição, quando não existirem mais colapsos, ***A*** será formado por apenas um nó e a floresta será composta por somente uma árvore (Figura 4.8).

O pseudo-código abaixo ilustra o algoritmo proposto considerando a hierarquia em vez do histórico:

*1° passo: constrói o espaço de colapsos eliminando células inativas //O(**n**)*
*2° passo: constrói o heap //O(**a**)*
*3° passo: constrói a floresta inicial //O(**n'**)*
4° passo: enquanto houver colapsos
{
*extrai o menor colapso //O(log**a**)*
*atualiza o grafo de vizinhança //O(**g**)*
*atualiza o heap //O(**g**log**a**)*
atualiza a floresta //O(1)
}

A complexidade final do algoritmo continua sendo O(***n'***log***n'***), apesar da alteração da complexidade do terceiro passo.

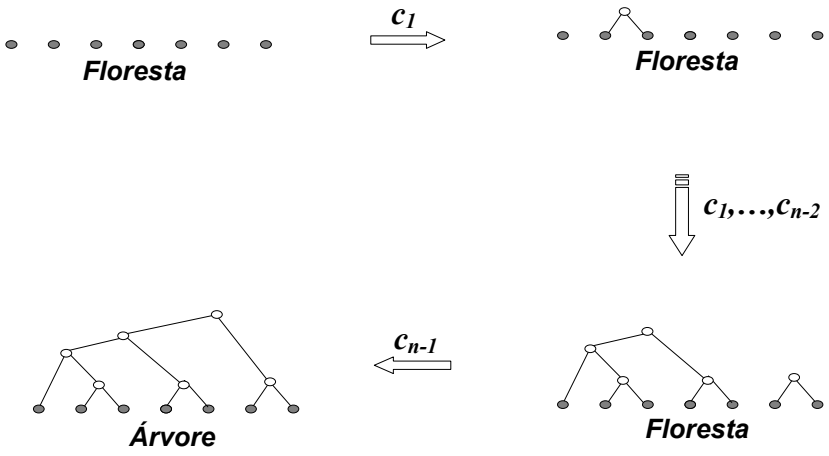

Figura 4.8 - Construção da hierarquia a cada iteração do algoritmo.

4.2 Características da Operação de Colapso

Tipicamente, a eficiência das operações incrementais é decisiva para o bom desempenho dos algoritmos gulosos de multi-resolução, pois elas realizam cálculos computacionalmente custosos [Gar99] que definem qual deve ser a nova geometria ótima que minimiza o erro segundo alguma norma pré-estabelecida. Como, na

maioria das vezes, a solução ótima é computacionalmente muito cara, heurísticas ou aproximações passam a ser utilizadas. Nessas heurísticas existe, geralmente, uma relação inversa entre tempo de processamento e qualidade do resultado obtido, ou seja, cálculos mais exatos, que produzem melhores resultados, são normalmente mais custosos.

4.2.1 Geometria e Valor de Propriedade

Visando alcançar uma boa eficiência computacional, a operação de colapso utiliza a vizinhança entre as células para calcular uma alternativa viável à macro-célula ótima. Isto é feito através da definição de três padrões fixos de colapso para o cálculo da macro-célula, um para cada direção ***I***, ***J*** e ***K*** da vizinhança entre células (Figura 4.9). Ou seja, se células são vizinhas em ***I***, por exemplo, é aplicado o padrão correspondente a ***I***. Os padrões são definidos através dos rótulos das faces (veja a Seção 3.2).

Para manter a estrutura de dados compacta, o colapso não gera novos vértices, aproveitando os vértices originais da malha de simulação. Portanto, a malha de multi-resolução tem o mesmo número de vértices da malha original. A Figura 4.9 ilustra o processo de formação da macro-célula nas 3 direções ***I***, ***J*** e ***K***. Neste processo, as faces extremas do intervalo ***I*** (ou ***J*** ou ***K***) são preservadas e as quatro novas faces da macro-célula são obtidas unindo-se as arestas correspondentes.

Neste momento, é importante ressaltar que há mais uma dimensão a ser levada em consideração pela operação de colapso: a propriedade das células definida em $\mathfrak{R}$. Assim, dada uma propriedade $\boldsymbol{i_k}$, o valor da propriedade da macro-célula, ***c,*** adotado neste trabalho, é a média ponderada dos valores de propriedade das células de entrada, ***a*** e ***b***, em relação ao seus volumes:

$$i_k(c) = \frac{i_k(a)\cdot v(a) + i_k(b)\cdot v(b)}{v(a) + v(b)}, \qquad \textbf{(4.1)}$$

onde $\boldsymbol{i_k(a)}$ e $\boldsymbol{i_k(b)}$ são os valores da ***k***-ésima propriedade nas células ***a*** e ***b*** e $\boldsymbol{v(a)}$ e $\boldsymbol{v(b)}$ são os volumes das células ***a*** e ***b***.

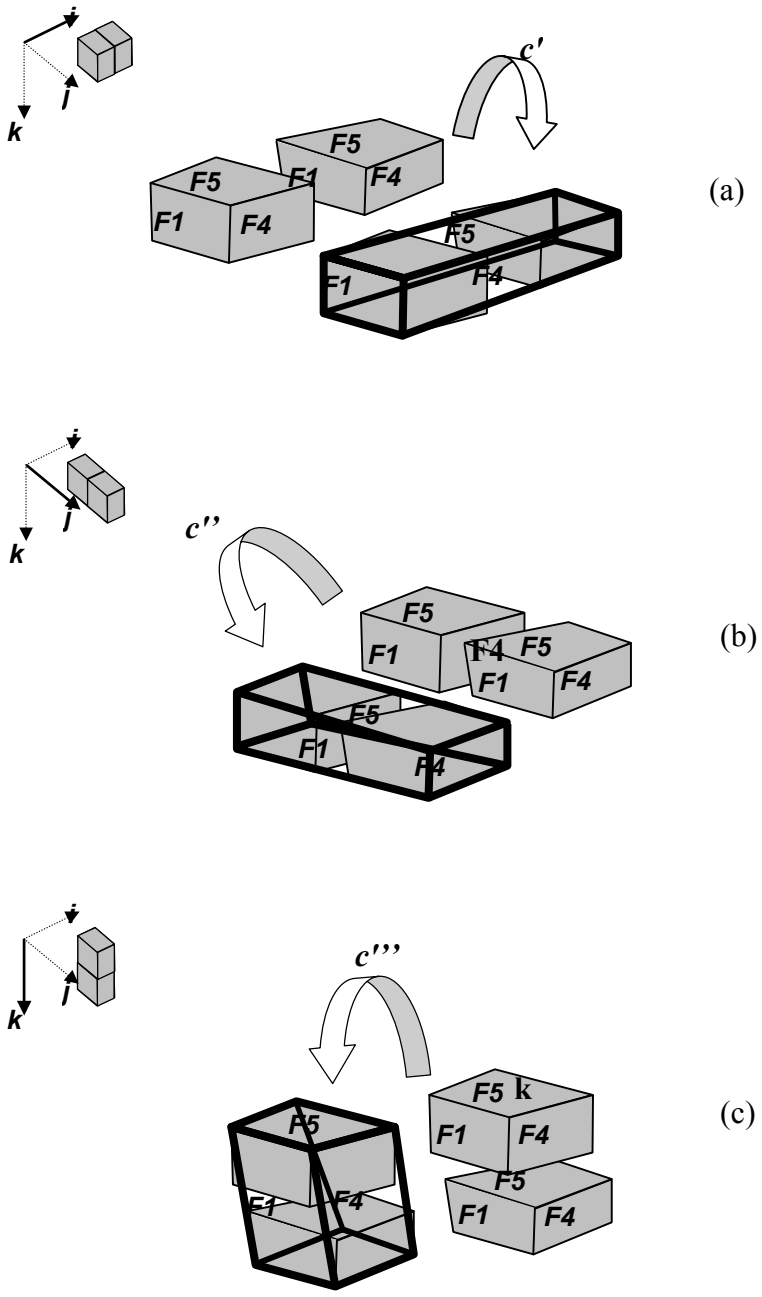

Figura 4.9 - Padrões para o cálculo da macro-célula baseado na vizinha do espaço paramétrico. Em (a), o padrão na direção *I*. Em (b), o padrão na direção *J*. Em (c), o padrão na direção *K*.

A equação (4.1) visa considerar características perceptuais no cálculo, uma vez que as células de maior volume possivelmente ocupem uma maior área no plano de projeção que forma a imagem.

4.2.2 Erros Geométrico e de Propriedade

Para cada macro-célula candidata a ser formada, é necessário calcular um custo levando-se em conta os erros geométrico e de propriedade. O erro pode ser computado através de alguma média ponderada entre os dois erros citados ou pela priorização de um erro em relação aos outros.

O erro geométrico entre duas células, ***a*** e ***b***, e uma macro-célula, ***c***, que as represente pode ser definido por uma norma semelhante à distância de Hausdorff [Pre85]. Ou seja, se ***S*** é o conjunto dos pontos da fronteira de ***a*** e ***b*** e ***T*** é o conjunto dos pontos da fronteira de ***c*,** então o erro, $\varepsilon_g(c)$, é definido como sendo:

$$\varepsilon_g(c)=\varepsilon(T,S)=\max(d(s,T),d(t,S)), \qquad \textbf{(4.2)}$$

onde ***d(s,T)*** é a distância máxima dos pontos da fronteira de ***S*** para a fronteira de ***T*** e ***d(t,S)*** é a distância máxima dos pontos da fronteira de ***T*** em relação à fronteira de ***S***. A escolha dos pontos de fronteira, e não do interior das células, baseia-se no interesse pela imagem gerada pela restituição (*rendering*) das faces do modelo. Ou seja, como somente as fronteiras das células são utilizadas na visualização, o algoritmo proposto se concentrou na diferença entre elas.

Para aumentar a eficiência do algoritmo proposto e tirar proveito das características especiais do modelo de RNPs, optou-se por aproximar ***d(s,T)*** e ***d(t,S)*** por duas distâncias fáceis de calcular. A primeira é estimada como sendo a maior distância dos vértices de ***S*** até as faces de ***T*** e a segunda como sendo a distância entre os vértices de ***S*** que foram eliminados na formação da macro-célula. A Figura 4.10 ilustra estas distâncias com setas em preto e azul, respectivamente. A justificativa para aproximar ***d(t,S)*** por esta distância vem do fato de que, quanto maior for o espaço entre as células (*gap*), maior será a distância dos pontos de ***T*** ao conjunto ***S***.

O erro de propriedade da macro-célula na propriedade i_k é dado pela função $\varepsilon i_k(T, S)$, que na sua forma reduzida é:

$$\varepsilon i_k(c) = \max\{|\, i_k(a) - i_k(c)\,|, |\, i_k(b) - i_k(c)\,|\}, \qquad \textbf{(4.3)}$$

Para considerar ***L*** propriedades no cálculo do erro pode-se utilizar a equação:

$$\varepsilon i(c)=\frac{1}{L}\sum_{l=0}^{L}\max\{|il(a)-il(c)|,|il(b)-il(c)|\}, \quad \textbf{(4.4)}$$

Tanto o erro geométrico quanto o erro de propriedade respeitam à norma $\boldsymbol{L_\infty}$ localmente e é importante observar que o valor de propriedade associado à macro-célula na equação (4.1) não minimiza a norma $\boldsymbol{L_\infty}$ utilizada na equação (4.3).

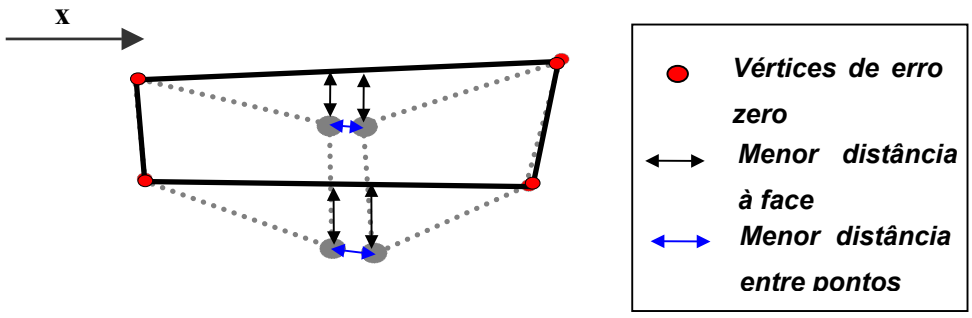

Figura 4.10 - Distâncias consideradas no cálculo do erro geométrico.

4.2.3 Tratamento de Múltiplas Propriedades

Uma vez que o modelo de um reservatório comumente possui dezenas de propriedades diferentes associadas, é inviável construir uma estrutura de MR para cada uma delas. Uma alternativa seria utilizar a equação (4.4) no cálculo de erro e considerar todas as propriedades numa única estrutura. Entretanto, além de aumentar o tempo gasto para a construção da hierarquia, esta solução não garante bons resultados.

Neste trabalho, optou-se por considerar apenas o erro geométrico na construção da estrutura MR. Os cálculos referentes às propriedades são feitos por demanda, de acordo com a necessidade de visualização. Percebe-se que estes cálculos não mais afetarão a estrutura da hierarquia, apenas adicionarão novos valores de propriedade e erro aos seus nós.

4.2.4 Propriedade de Ordenação Parcial

Os cálculos do erro geométrico e de propriedade utilizados pela operação de colapso garantem limites de erro segundo a norma L_∞ apenas entre aproximações do reservatório geradas por cortes na hierarquia que possuam, no máximo, um nível de diferença.

Analisando a versão unidimensional do problema (hierarquia de arestas para linhas poligonais), pode-se perceber as implicações deste fato (Figura 4.11). O erro geométrico $\varepsilon_g(l_7)$ é o erro real entre $\{l_7\}$ e $\{l_5, l_6\}$. Entretanto, na ilustração, fica evidente que o erro geométrico real entre $\{l_7\}$ e as folhas $\{l_1, l_2, l_3, l_4\}$ é maior do que $\varepsilon_g(l_7)$. Ou seja, a estrutura de multi-resolução exemplificada é incapaz de garantir um limite de erro em relação ao modelo original.

Isto seria exatamente o que aconteceria com a hierarquia de células se fossem utilizados diretamente os erros da operação de colapso. Refazer os cálculos da operação de forma que eles sejam realizados diretamente sobre o modelo original é demasiadamente custoso e violaria a característica de visualização que requer baixo tempo de pré-processamento (veja a Seção 3.3). Alternativamente, poderia-se pensar em utilizar para o erro de uma macro-célula o maior valor dentre o erro calculado pela operação de colapso e a soma dos erros das células filhas. Assim, sendo a e b as células de entrada de um colapso e c a macro-célula gerada, tem-se:

$$\varepsilon'_g(c)=\max\{\varepsilon_g(c),\varepsilon_g(a)+\varepsilon_g(b)\}.$$

Provavelmente, este novo erro seria um limite superior para o erro real. Entretanto, o fato desse valor ser utilizado pelo *heap* para avaliar o custo de um colapso durante a construção da estrutura de MR torna essa super-estimação inadequada, pois colapsos de qualidades distintas poderiam ser classificados (estimados) da mesma forma.

A solução utilizada neste trabalho consiste em considerar o erro como o máximo entre o erro da operação de colapso e os erros das células de entrada da operação:

$$\varepsilon'_g(c)=\max\{\varepsilon_g(c),\varepsilon_g(a),\varepsilon_g(b)\}, \qquad \textbf{(4.5)}$$

Esta solução busca uma medição mais realista do erro sem, com isto, aumentar o custo dos cálculos, mas não garante um limite de erro em relação ao modelo original. Espera-se que, no caso de RNPs, a boa coplanaridade entre as células dos reservatórios e o caráter volumétrico da operação de colapso façam com que os resultados sejam satisfatórios.

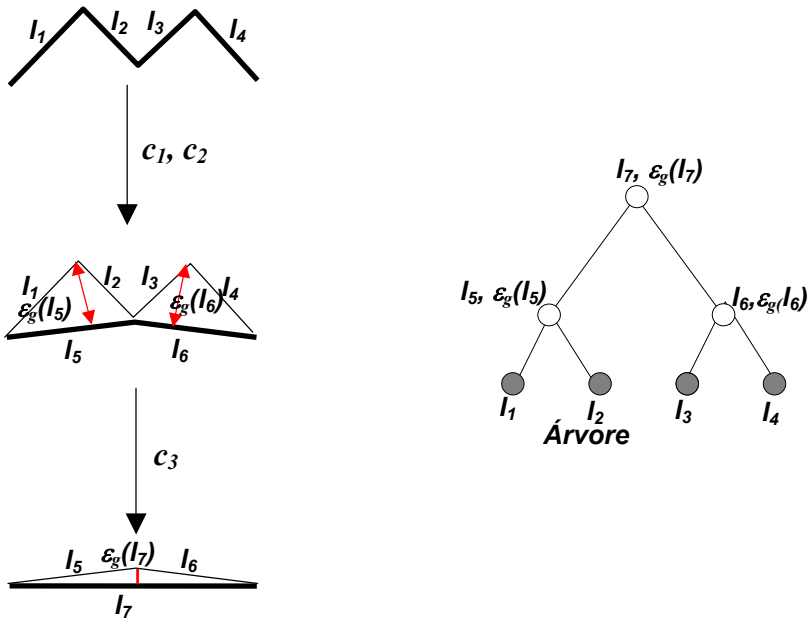

Figura 4.11 - Exemplo unidimensional: colapsos de arestas. O erro, $\varepsilon_g(I_7)$, associado ao colapso c_5 é menor do que os erros, $\varepsilon_g(I_5)$ e $\varepsilon_g(I_6)$, associados aos colapsos c_1 e c_2.

A utilização da equação (4.5) faz com que a estrutura de MR tenha uma propriedade de ordenação parcial decrescente em relação ao erro desejável para estruturas de multi-resolução.

Sejam ***a*** e ***b*** as células de entrada de um colapso e ***c*** a macro-célula gerada, tem-se que:

$$\varepsilon'_g(c) >= \varepsilon'_g(a) \quad \wedge \quad \varepsilon'_g(c) >= \varepsilon'_g(b).$$

E, portanto:

> "*o erro associado a um nó interno da hierarquia é sempre maior ou igual aos erros associados aos seus nós filhos*", **P.4.2**

O mesmo raciocínio pode ser aplicado para o caso do erro de propriedade. Assim, tem-se a equação:

$$\varepsilon i'_I(c)=\max\{\varepsilon i_I(c),\varepsilon i_I(a),\varepsilon i_I(b)\}, \qquad \textbf{(4.6)}$$

Com isso:

$$\varepsilon i'_I(c)>=\varepsilon i'_I(a) \quad \wedge \quad \varepsilon i'_I(c)>=\varepsilon i'_I(b).$$

E, portanto, levando em consideração uma propriedade específica:

"*o erro de propriedade associado a um nó interno da hierarquia é sempre maior ou igual aos erros de propriedade associados aos seus nós filhos*". ***P.4.3***

4.3 Propriedades de Extração

Em relação à estrutura combinatorial utilizada pelo MMR proposto, é válido fazer algumas observações importantes. Em primeiro lugar, nota-se que qualquer corte na hierarquia de MR gerado por um caminhamento pré-fixado corresponde a uma aproximação do reservatório que abrange todo o seu domínio sem que haja porções do domínio cobertas por mais de uma célula (Figura 4.12). Uma vez que são permitidos modelos não-conformes, o fato anterior constitui a regra básica de extração de modelos do MMR proposto.

Considerando isso, pode-se atribuir à hierarquia de células algumas propriedades que são bastante desejáveis a estruturas de MR:

- alto poder de expressão – o número possível de aproximações corresponde ao número possível de cortes, que é significativo uma vez que a estrutura é binária, permitindo uma grande variação de resolução ao longo do domínio do reservatório;

- extração ótima proporcional ao tamanho da aproximação – cada corte na árvore está associado a uma sub-árvore, como ilustrado na Figura 4.13. Somente os nós da sub-árvore são visitados pelo caminhamento que gera a aproximação. Uma vez que o número de nós de uma árvore binária é aproximadamente o dobro do seu número de folhas, tem-se que a complexidade de extração é linear em relação ao número de células da aproximação.

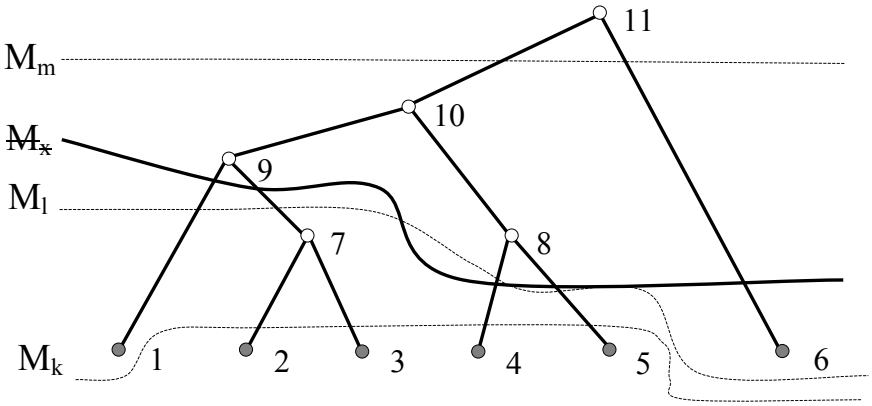

Figura 4.12 – Corte na árvore. O modelo gerado por um corte corresponde às folhas da sub-árvore que está acima do corte. Os cortes M_k, M_l e M_m formam três modelos válidos que abrangem todo o domínio do objeto. O corte inválido ~~M_x~~ apresenta redundância e não pertence ao conjunto dos cortes gerados pelo caminhamento pré-fixado.

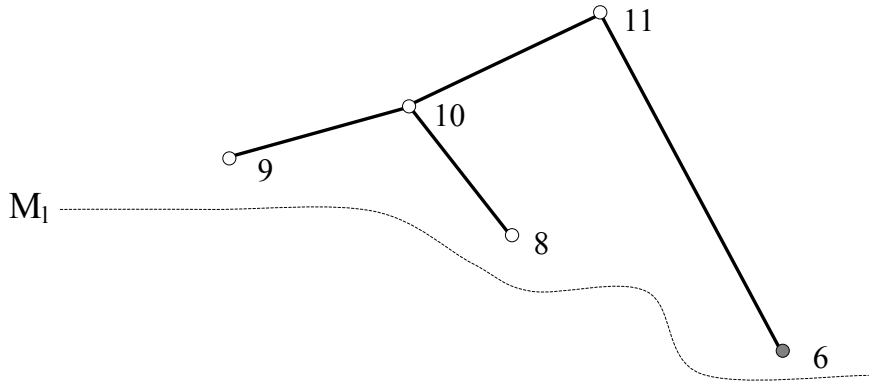

Figura 4.13 – Sub-árvore associada ao corte M_l. A árvore é a mesma da Figura 4.12.