2
Trabalhos relacionados

Neste capítulo, são revistas as principais questões relativas à representação distribuída de malhas de elementos finitos, e os trabalhos relacionados propostos na literatura que visam tratá-las.

Uma abordagem comum utilizada na paralelização de aplicações que executam uma mesma operação sobre um determinado volume de dados é o modelo de *paralelismo de dados* (Foster, 1995; Mattson et al., 2004). Esse modelo se aplica a modelos de elementos finitos, nos quais operações são realizadas sobre os elementos e nós de uma malha. A malha é decomposta em um conjunto de partições, e a cada partição é atribuído um subconjunto dos elementos e nós da malha original. Uma partição representa, assim, uma unidade local de processamento, que pode ser associada a um determinado processador para ser executada concorrentemente com outras partições; em geral, um processador é responsável por uma ou mais partições. Dessa forma, a malha global passa a ser representada de forma distribuída, pelo conjunto de partições associadas a um grupo de processadores.

A representação distribuída de malhas de elementos finitos (Remacle et al., 2002; Seol & Shephard, 2006; Lawlor et al., 2006) oferece uma ferramenta para a execução de análises paralelas visando a solução de problemas maiores e/ou a redução do tempo total de simulação. Por outro lado, também introduz questões que devem ser consideradas a fim de que a análise possa ser realizada de forma eficiente. Algumas dessas questões são apresentadas nas seções a seguir. Na Seção 2.4, diversos sistemas presentes na literatura para a representação de malhas distribuídas dinâmicas são brevemente discutidos.

2.1.
Atualização de dados compartilhados por múltiplas partições

Uma questão importante relativa à representação de malhas distribuídas refere-se à manutenção da consistência de dados entre as partições de malha; em especial, consideram-se os dados correspondentes a entidades localizadas próximas às fronteiras das partições.

Computações de resultados de simulação para uma entidade topológica do modelo de elementos finitos muitas vezes dependem de dados associados às entidades adjacentes a ela. Porém, uma entidade localizada na fronteira de uma partição pode ser adjacente a entidades localizadas em outras partições. No exemplo da Figura 3, o nó destacado é compartilhado por elementos de duas partições de malha diferentes. Se a computação do valor do nó depender dos valores dos elementos adjacentes, estes devem ser obtidos a partir das partições correspondentes. Isso requer uma forma de comunicação entre as partições envolvidas. Para que o valor do nó possa ser utilizado pelas partições que o compartilham, ele deve ser atualizado de maneira consistente entre elas.

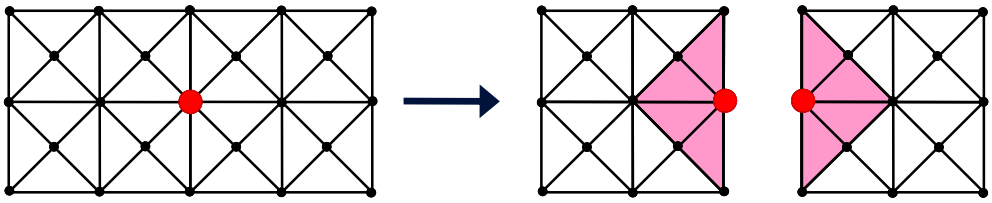

Figura 3 – Um nó compartilhado por elementos pertencentes a duas partições de malha diferentes encontra-se destacado. O valor do nó deve ser atualizado consistentemente entre as partições.

Em arquiteturas de *memória compartilhada* (Foster, 1995), todos os processadores compartilham um mesmo espaço de memória virtual. Isso permite que a partições de malha correspondentes se comuniquem de maneira assíncrona, através de operações de escrita e leitura de dados em memória. Por sua vez, a atualização de regiões de memória compartilhadas pode ser feita de forma consistente por meio de mecanismos de acesso exclusivo, como *travas de acesso* (*locks*) e *semáforos* (Andrews, 2000). No contexto de estruturas de dados topológicas, Waltz (2002) apresenta algoritmos paralelos para o acesso a entidades topológicas, que podem ser utilizados em simulações baseadas no MEF.

Em ambientes paralelos baseados na arquitetura de *memória distribuída* (Foster, 1995), nós de processamento independentes (contendo um ou mais processadores) são interconectados por uma infraestrutura de rede (em geral, de alto desempenho). Neste caso, o espaço de memória de um nó não é compartilhado com os outros, e a transmissão de dados entre nós é tipicamente realizada através do envio de mensagens através da rede. O acesso indireto a dados remotos geralmente representa um aumento do custo da comunicação

entre partições. Por outro lado, os ambientes de memória distribuída têm se mostrado escaláveis, permitindo que o número de processadores alcance dezenas ou mesmo centenas de milhares, o que dificilmente é obtido apenas com arquiteturas de memória compartilhada.

Uma solução comumente empregada em ambientes de memória distribuída para se garantir acesso consistente e eficiente a dados remotos pelas partições de uma malha consiste na introdução de uma *camada fantasma* ao redor de cada partição de malha. Convencionalmente, essa camada é composta por cópias locais *não editáveis* de entidades topológicas remotas que são adjacentes às entidades de uma partição. Dessa forma, os valores das entidades adjacentes podem ser acessados pela partição durante uma computação de maneira local e transparente. As *entidades fantasmas*, em geral, são atualizadas todas ao mesmo tempo, ao final de um passo de simulação, ou em qualquer outro momento em que isso seja necessário. Assim, a frequência de requisições de dados a outras partições é significativamente reduzida. Com a atualização de dados de uma entidade topológica realizada por apenas uma partição, considerada a *proprietária* da entidade, garante-se a consistência dos dados.

Uma variação da solução anterior é adotada pelo *framework* ParFUM (Lawlor et al., 2006). O conceito de *Multiphase Shared Arrays (MSA)* (DeSouza & Kalé, 2004) é utilizado para manter dados de nós (como na Figura 3) compartilhados por partições de malha distintas atualizados de maneira assíncrona. Para isso, um vetor distribuído é responsável por intermediar a comunicação entre partições. Em uma primeira etapa, o acesso ao vetor é feito através de um modo de *acumulação*, e os dados de cada partição são atribuídos às posições dos nós correspondentes no vetor, que os soma (*acumula*) automaticamente. Na segunda etapa, o modo de acesso é alterado para *leitura*, e cada partição obtém os dados dos nós de forma consistente. Os valores dos outros elementos e nós da camada fantasma são atualizados conjuntamente, por meio de uma chamada de função paralela coletiva específica, ao final de cada passo de simulação.

2.2. Particionamento da malha de elementos finitos

O particionamento da malha de elementos finitos exerce papel fundamental na eficiência da simulação paralela, uma vez que o custo total de um passo de simulação pode ser determinado pelo da partição de maior custo.

Em geral, o custo (ou tempo) de computação de uma partição é proporcional ao número de elementos e/ou nós que ela contém. Assim, é desejável que essas entidades encontrem-se balanceadas entre as partições, de forma que tempos ociosos sejam minimizados, e o tempo total de simulação reduzido.

Um ponto importante a ser considerado durante o particionamento de malhas é o custo de comunicação entre as partições. Devido à necessidade de comunicação para a atualização dos elementos e nós próximos às fronteiras de partições, é desejável agrupá-los de modo a se minimizar o tamanho das fronteiras compartilhadas com outras partições. Embora, em geral, uma partição se comunique apenas com um pequeno número de partições vizinhas a ela, o custo da comunicação ainda pode ser significativo, especialmente em partições de tamanho reduzido.

Dois sistemas populares para particionamento de malhas são METIS (Karypis & Kumar, 1995, 1998a) e sua versão paralela ParMETIS (Karypis & Kumar, 1998b). Esses sistemas utilizam partição de grafos para decompor grandes malhas de elementos finitos de forma eficiente, e buscam, ao mesmo tempo, balancear o número de elementos de cada partição e minimizar as fronteiras compartilhadas por diferentes partições. Nesta pesquisa, o sistema METIS é empregado para particionar as malhas usadas em alguns dos experimentos computacionais realizados.

2.3. Malhas adaptativas

Em análises adaptativas de elementos finitos, a malha é modificada dinamicamente ao longo do processo de simulação numérica. Dessa forma, o suporte topológico de malha deve fornecer operadores de edição apropriados. No contexto de simulações dinâmicas de fraturas baseadas no modelo coesivo extrínseco (Camacho & Ortiz, 1996; Ortiz & Pandolfi, 1999; Park et al., 2009), requer-se suporte para a representação e inserção adaptativa de elementos coesivos. Os operadores topológicos de edição de malha devem ser capazes de manter a topologia da malha consistente após cada operação, o que apresenta desafios adicionais à implementação em ambientes paralelos.

Para que o comportamento não-linear de regiões ao redor de pontas de fraturas (Zhang, 2007) seja corretamente capturado, um grande grau de precisão numérica é, em geral, requerido. Uma forma de se aumentar a precisão numérica do MEF consiste no refinamento de elementos da malha (*h-refinement*)

(Zienkiewicz et al., 2005). Contudo, uma vez que um alto nível de refinamento não é necessário em regiões distantes de fraturas, é interessante que o nível de refinamento se adapte ao grau de precisão exigido em cada região de malha. Simulações de fraturas sequenciais que empregam refinamento e simplificação adaptativa, e outras otimizações geométricas e topológicas de malha, com base na estrutura de dados topológica TopS, foram recentemente apresentadas por Paulino, Park, Celes & Espinha (2010) e Park, Paulino, Celes & Espinha (2011).

Em análises onde a malha distribuída permanece estática ao longo da simulação, o balanceamento da carga dos processadores pode feito apenas uma vez, durante o particionamento inicial da malha – sistemas como METIS (Karypis & Kumar, 1995, 1998a) podem ser empregados para isso. Em análises adaptativas, por outro lado, conforme se modifica a malha, o número de elementos e nós em um processador pode divergir significativamente de outros, refletindo-se nas cargas de computação correspondentes. Dessa forma, o balanceamento de carga dinâmico (Hendrickson & Devine, 2000; Ozturan, 1995; Devine et al., 2005) torna-se importante para a manutenção do desempenho da simulação. A biblioteca Zoltan (Devine et al., 2002) oferece utilitários para o gerenciamento de malhas dinâmicas distribuídas, como: partitionamento, balanceamento de carga e procedimentos para comunicação entre partições. O *framework* Charm++ (Kalé & Krishnan, 1993, 1996), para o desenvolvimento de programas paralelos, oferece suporte ao balanceamento de carga automático e à definição de novos balanceadores de carga pela aplicação cliente.

Esta pesquisa foca na representação e criação de fraturas dinâmicas. Para isso, assume-se que os elementos da malha encontram-se apropriadamente refinados e a malha distribuída corretamente balanceada. Uma vez que o número de elementos coesivos e o custo correspondente são tipicamente muito menores que o de elementos volumétricos, considera-se que o balanço de cargas não é significativamente alterado com a inserção de novos elementos.

2.4. Estruturas de dados topológicas para malhas distribuídas dinâmicas

Diversos sistemas paralelos com suporte à representação distribuída de malhas dinâmicas de elementos finitos não-estruturadas são encontrados na literatura (Remacle et al., 2002; Lawlor et al., 2006; Seol & Shephard, 2006; Ozturan et al., 1994; Kirk et al., 2006; Stewart & Edwards, 2004). Embora esses sistemas sejam capazes de representar malhas gerais de elementos finitos, um

suporte apropriado para fraturas *extrínsecas* tridimensionais não é apresentado. Os principais sistemas existentes são brevemente discutidos a seguir.

2.4.1. MDB/PMDB

O sistema *Parallel Mesh Database* (PMDB) (Ozturan et al., 1994; Ozturan, 1995) oferece operações para a manipulação de malhas distribuídas gerais. Esse sistema é implementado com base na estrutura de dados sequencial *Mesh Database* (MDB) (Beall & Shephard, 1997). São representadas entidades topológicas como: *região*, *face*, *aresta* e *vértice*. Cada região é associada a um único processador, enquanto que faces, arestas e vértices são duplicados em cada processador que contém regiões adjacentes a essas entidades. Cada entidade duplicada mantém uma lista de suas cópias em outras partições. Em PMDB, o conceito de pertinência única de entidades é implementado. Dessa forma, apenas uma partição pode ser considerada a proprietária de uma entidade topológica. A partição proprietária é determinada pelo valor mínimo da tupla (p_i, e_i) entre a lista de usos da entidade, onde p_i é o identificador (*id*) da partição e e_i é o identificador (*id*) da cópia da entidade na partição. Entre os operadores de malha oferecidos encontram-se a inserção e remoção de entidades e a obtenção de informações de adjacência. Também são oferecidas funcionalidades para particionamento de malhas e balanceamento de carga.

2.4.2. AOMD/PAOMD

O sistema *Parallel Algorithm Oriented Mesh Database* (PAOMD) (Remacle et al., 2002) estende a versão sequencial anterior chamada *Algorithm Oriented Mesh Database* (AOMD) (Remacle et al., 2000; Remacle & Shephard, 2003) com suporte para malhas distribuídas. É fornecido um arcabouço (*framework*) para o gerenciamento de malhas distribuídas genéricas, no qual a representação de malhas pode ser adaptada a diferentes aplicações. Várias entidades topológicas (vértices, arestas, faces e regiões), além de relações de adjacência entre elas, podem ser representadas; a aplicação cliente determina quais as necessárias.

Entidades topológicas, com exceção de vértices, são definidas e representadas por conjuntos de entidades de dimensão inferior e sua ordenação local correspondente. PAOMD requer que um identificador global único (*global id*) seja associado a cada vértice, e que uma entidade possa ser identificada pela

sua lista de vértices. Identificadores globais, entretanto, podem ser inconvenientes à representação de malhas adaptativas, pois devem ser corretamente mantidos entre as partições de malha. Em relação à representação de elementos coesivos, a definição de entidades a partir de seus vértices apresenta ambiguidades, já que esses elementos consistem de duas facetas (faces, em 3D, ou arestas, em 2D) que são diferentes, mas podem possuir os mesmos vértices (conforme Seção 2.6.2). Nesse caso, a representação por vértices não é suficiente para se distinguir entre entidades diferentes.

Assim como em PMDB, cada partição é associada a um processador, e a malha local é representada por uma malha sequencial AOMD. Entidades classificadas nas fronteiras de uma partição devem existir na estrutura de dados paralela e ser compartilhadas com as partições vizinhas. A conexão de uma entidade com suas cópias em outras partições é feita por mensagem enviada a todas as partições sempre que a malha é modificada. A mensagem contém o endereço local da entidade e a lista dos identificadores (*ids*) dos seus vértices. Procedimentos para malhas adaptativas, balanceamento de carga dinâmico e migração de entidades entre partições são fornecidos pelo sistema.

2.4.3. FMDB

A infra-estrutura de malha paralela *Flexible distributed Mesh DataBase* (FMDB) (Seol & Shephard, 2006) permite a representação de modelos genéricos não-manifold. Assim com o sistema PAOMD, a aplicação cliente pode configurar os tipos de entidades topológicas necessários. Cada partição é representada por uma malha sequencial, porém as entidades nas fronteiras da malha são tratadas de forma diferente. Elas são duplicadas em todas as partições onde são requeridas à computação de relações de adjacências. Todavia, apenas uma partição pode ser considerada a *proprietária* de uma entidade topológica. Essa partição é a que contém o menor número de objetos entre as partições nas quais a entidade é duplicada, a fim de evitar que a carga entre os processadores se torne desbalanceada quando a malha é modificada. Um algoritmo para migração eficiente de entidades topológicas entre partições é oferecido por FMDB.

2.4.4. LibMesh

O sistema LibMesh (Kirk et al., 2006) trata de aspectos independentes da física de análises de elementos finitos. São oferecidos um suporte à representação de malha distribuída, funções para adaptação da malha e interfaces para outros sistemas comumente utilizados em aplicações baseadas no MEF. A representação de malha é baseada na estrutura de dados tradicional de elementos e nós, e um identificador (*id*) global único é atribuído a cada elemento e nó da malha. Elementos também possuem um identificador do processador ao qual estão atribuídos, referências para os nós incidentes (conectividade nodal) e aos outros elementos adjacentes às suas faces (elementos vizinhos). Entretanto, uma cópia completa da malha é armazenada em cada processador, apesar da decomposição lógica em partições, o que limita o uso desse sistema em simulações de larga escala. De acordo com Kirk et al. (2006), uma implementação completamente distribuída está sendo considerada.

2.4.5. SIERRA

O sistema SIERRA (Stewart & Edwards, 2004) oferece diversas ferramentas para o desenvolvimento de aplicações para análises mecânicas. Fazem parte de SIERRA uma estrutura de dados topológica distribuída para a representação de malhas, suporte a adaptação de malha e balanceamento de carga, e uma interface a bibliotecas de solução de sistemas lineares (*linear solvers*), entre outras funcionalidades. A estrutura topológica representa entidades como: *nó*, *aresta*, *face* e *elemento*. Elementos, faces e arestas são definidos por um conjunto de nós que são também vértices. Cada uma dessas entidades pode se conectar a entidades de tipos diferentes, e as conexões entre entidades podem ser configuradas pela aplicação cliente. Por exemplo, relações de adjacência podem ser criadas para se obterem as faces de todos os elementos *fantasmas* definidos nas fronteiras de partições. Assim como outras estruturas de dados, entidades nas fronteiras entre partições podem ser compartilhadas entre as partições, porém apenas uma partição é a *proprietária* da entidade. Entidades são univocamente identificadas pela tupla (*tipo, id*), onde *tipo* é o tipo da entidade (i.e., nó, elemento, etc.) e *id* é um valor inteiro único entre todas as entidades de um mesmo tipo. Infelizmente, SIERRA não se

encontra publicamente disponível, e poucas informações sobre esse sistema podem ser encontradas na literatura.

2.4.6. ParFUM

O sistema ParFUM (Lawlor et al., 2006) oferece suporte à representação distribuída de malhas não-estruturadas. Esse sistema é implementado com base no *framework* paralelo Charm++ (Kalé & Krishnan, 1993, 1996) e AMPI (Huang et al., 2003), uma implementação da especificação de interface de envio de mensagens MPI (MPI Forum, 2010) sobre Charm++.

A estrutura de dados topológica é capaz de representar as entidades *elemento* e *nó*, além de informações adicionais de adjacência, como as conectividades *nó-a-nó*, *nó-a-elemento* e *elemento-a-elemento*. Uma partição de malha é chamada *chunk*, e é geralmente associada a um único processador (ou processo MPI). Por outro lado, um processador pode ter diversos *chunks* associados a ele. A comunicação entre *chunks* ocorre de maneira implícita, através de duas classes de entidades especiais (Figura 4): nós *compartilhados* (*shared nodes*) e nós e elementos *fantasmas (ghost nodes/elements)*. Cada elemento é associado a um único *chunk*, enquanto que nós podem ser compartilhados por elementos de *chunks* diferentes. Os nós classificados como *compartilhados* são duplicados em cada *chunk* em que são usados. Durante a simulação numérica, entidades nas fronteiras de uma partição podem precisar de informações de entidades vizinhas representadas em outras partições. Para isso, são criadas camadas fantasmas (*ghost layers*) de elementos ao redor de cada *chunk*. As entidades da camada fantasma (nós e elementos *fantasmas*) são disponibilizadas apenas para leitura (*read-only*), segundo a maneira convencional, não podendo ser editadas.

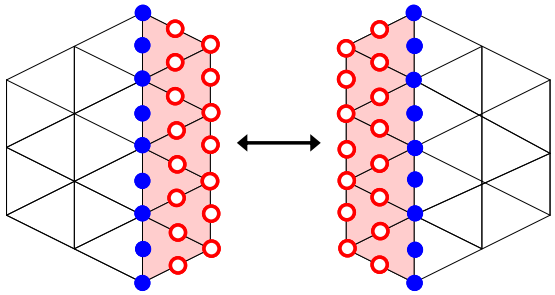

Figura 4 – Partições com camada *fantasma*. Os nós compartilhados são desenhados preenchidos. Os elementos e nós da camada fantasma também encontram-se destacados.

Em cada *chunk*, um índice local (relativo ao *chunk*) é atribuído às entidades presentes. Se dois *chunks* compartilham nós, então eles mantém uma lista mútua de índices locais de nós compartilhados, consistentemente ordenados. Com isso, o mapeamento do índice local de um nó de um *chunk* para o outro é feito de maneira direta. No caso de entidades fantasmas, o *chunk* que possui a entidade real mantém uma lista de mapeamento (*sendghost list*), destinada ao envio de dados às partições que possuem cópias da entidade. Um *chunk* contendo uma entidade fantasma mantém uma lista de mapeamento reversa (*receivenode list*). Para sincronizar os dados de entidades da camada fantasma com as entidades reais correspondentes, são fornecidas funções paralelas coletivas.

São suportadas operações incrementais de modificação topológica de malhas bidimensionais. A modificação de malhas em paralelo é baseada em operadores atômicos (Choudhury, 2006), com a atomicidade garantida por requisições de acesso exclusivo (*travas de acesso*, ou *locks*) aos nós envolvidos. Para que um elemento na fronteria de um *chunk* seja modificado, o acesso exclusivo aos respectivos nós são requisitadas. Uma vez que a partição detém o acesso exclusivo a todos os nós, o elemento é alterado localmente, e os *chunks* se comunicam para sincronizar as informações topológicas com os *chunks* vizinhos. Finalmente, as travas obtidas são liberadas e podem ser requisitadas por outros *chunks* interessados. Isso permite que algoritmos de adaptação de malha utilizem operadores sequenciais, sem a necessidade de sincronização explícita. Por outro lado, o custo para se obter acesso exclusivo aos nós, e o número de mensagens trocadas entre partições, pode ser significante. Choudhury (2006) reduz esses custos com a alocação de múltiplos *chunks* de menor tamanho em cada processador, de forma a aumentar a concorrência e reduzir tempos ociosos (*idle time*) de processadores.

Resultados de simulação de fraturas com base em ParFUM são encontrados na literatura (Mangala et al., 2008). Entretanto, o suporte a representação de fraturas se restringe a malhas triangulares bidimensionais, ou utiliza elementos coesivos pré-inseridos em toda a malha, e que são ativados sob demanda. Um suporte topológico em paralelo, completo e geral, para a representação e inserção dinâmica de elementos coesivos *realmente extrínsecos*, tanto em malhas bidimensionais (2D) como tridimensionais (3D), ainda é necessário.

2.5. Simulações paralelas de fraturas coesivas extrínsecas

Nesta seção, são revistos trabalhos propostos na literatura que abordam simulações em paralelo de fraturas baseadas em modelos de zona coesiva extrínsecos. A revisão é feita sob o ponto de vista da representação de dados topológicos, conforme o foco desta pesquisa.

Quando uma abordagem intrínseca (Xu & Needleman, 1994) é utilizada, elementos coesivos são criados ao longo de todas as interfaces entre elementos volumétricos da malha, antes do início da simulação. Dessa forma, os elementos estão presentes durante todo o processo de simulação, contribuindo para os resultados calculados. Como não são necessárias modificações topológicas da malha após o início da simulação, essa abordagem é inerentemente escalável. As questões tratadas pela representação de malha distribuída consistem apenas no particionamento da malha original e a comunicação entre as partições.

Modelos extrínsecos, no entanto, introduzem desafios adicionais. Elementos coesivos devem ser inseridos sob demanda quando um critério de fratura é alcançado. Isso faz com que a representação de malha torne-se mais complexa, uma vez que a topologia da malha se modifica durante a simulação. As mudanças ocorridas localmente em cada partição devem ser propagadas para as partições vizinhas, de forma consistente e eficiente. Alguns algoritmos para a inserção dinâmica de elementos coesivos existem na literatura (Pandolfi & Ortiz, 1998, 2002; Mota et al., 2008; Paulino et al., 2008). Porém, devido às dificuldades de paralelização de modelos extrínsecos, em especial para malhas tridimensionais, poucas soluções em paralelo foram propostas.

Duas abordagens recentes contornam as dificuldades relacionadas à representação de fraturas utilizando elementos coesivos pré-inseridos. Dooley et al. (2009) empregam uma estratégia baseada na ativação de elementos, implementada para malhas bidimensionais. Com isso, elementos coesivos existem em todas as interfaces entre elementos volumétricos da malha inicial, mas permanecem inativos até que o critério de fratura seja alcançado nas interfaces correspondentes. Nesse momento, a lei coesiva entra em efeito, e o elemento torna-se parte do processo de simulação. Do ponto de vista da topologia da malha, essa abordagem equivale ao modelo intrínseco, no sentido em que elementos coesivos são pré-inseridos em todas as interfaces onde fraturas possam ocorrer. A diferença é que elementos coesivos são ativados na simulação apenas quando necessários.

Como a topologia da malha não se altera durante a simulação, comunicações entre partições não são necessárias para propagar modificações topológicas às partições vizinhas. Isso faz com que essa abordagem se torne escalável, assim como os modelos intrínsecos. Por outro lado, também introduz dificuldades que devem ser contornadas. Durante a construção da malha distribuída, todo nó localizado em fronteiras compartilhadas por duas ou mais partições é replicado tantas vezes quanto o número de elementos volumétricos adjacentes, de forma que os elementos coesivos possam ser criados. Assim, mesmo quando nenhum elemento coesivo está ativo na simulação, os elementos e as cópias dos nós correspondentes existem na topologia da malha de cada partição. Isso introduz um custo adicional para a representação de malha, em especial em aplicações com um número pequeno de interfaces fraturadas, devido à presença de nós e elementos desnecessários à simulação.

Além disso, a representação topológica real da malha não corresponde à efetivamente utilizada na simulação. Dessa forma, a aplicação cliente, ou uma camada intermediária adicional, deve ser responsável por prover o acesso consistente aos nós replicados. Enquanto um elemento coesivo não está ativo, cada par de nós compartilhados entre as duas arestas do elemento coesivo bidimensional (ou faces de um elemento tridimensional) deve corresponder a um único nó regular no contexto da simulação, embora os nós sejam representados como entidades topológicas diferentes. Para contornar as inconsistências resultantes de duplicações nodais desnecessárias, e assegurar a continuidade de malha na simulação, um nó aleatório é escolhido entre as múltiplas cópias, como *nó raiz representativo* ("*root node*"). A diferença entre a representação topológica da malha e a usada pela simulação também afeta o acesso a relações topológicas entre entidades da malha. O acesso a essas relações passa a não ter um tratamento uniforme, uma vez que dois elementos volumétricos que deveriam ser adjacentes na simulação encontram-se separados por um elemento coesivo na representação da malha.

A implementação da abordagem de Dooley et al. (2009) baseia-se no sistema ParFUM (Lawlor et al., 2006). O código da simulação paralela é dividido em duas rotinas principais: *init()* e *driver()* (Dooley et al., 2009). Em *init()*, a malha completa é carregada em um único processador. Após o término de *init()*, o sistema particiona a malha e cria a infraestrutura de comunicação entre partições. A simulação numérica é realizada pela rotina *driver()*, executada em paralelo por cada partição de malha. Com base em ParFUM, a comunicação entre partições é feita através de uma camada fantasma, com cópias não

editáveis de elementos e nós de partições vizinhas. A camada é criada ao redor das fronteiras das partições. Resultados de simulações dinâmicas de fraturas são apresentados para malhas de triângulos, com até 1,2 milhões de elementos e 512 processadores. Um trabalho complementar de Mangala et al. (2008) aborda o refinamento e simplificação adaptativo de malhas bidimensionais.

A abordagem proposta por Radovitzky et al. (2011) para a paralelização de simulações de fraturas extrínsecas é baseada na adaptação dos modelos numéricos utilizados. Dessa forma, uma formulação descontínua de Galerkin (*Discontinuous Galerkin - DG*) (Noels & Radovitzky, 2006, 2008) do problema contínuo é combinada com um modelo coesivo de fratura. Assim como a abordagem de Dooley et al. (2009), elementos coesivos são pré-inseridos em todas as interfaces entre elementos volumétricos, durante a construção da malha inicial. Termos adicionais introduzidos à formulação do problema, devidos à formulação descontínua de Galerkin, garantem a consistência da simulação na ausência de fraturas. Quando o critério de fratura é alcançado para uma determinada interface entre elementos volumétricos, a lei coesiva entra em efeito, substituindo os termos da formulação descontínua de Galerkin. Com isso, elementos coesivos permanecem na malha de forma consistente com a simulação numérica, mesmo na ausência de fraturas, ao contrário da abordagem anterior. A sincronização de atributos de simulação é feita utilizando-se uma operação paralela de redução (Quinn, 2004) para somar os resultados dos nós das fronteiras compartilhadas entre partições. Essa abordagem é escalável, de forma similar à anterior. Resultados para simulações de propagação de ondas (*wave propagation*) e fragmentação dinâmica são apresentados para malhas de tetraedros quadráticos (Bathe, 1996) com 103 milhões de elementos e 4096 processadores.

2.6. TopS

A estrutura de dados topológica *TopS*, para representação sequencial de malhas de elementos finitos, foi proposta por Celes, Paulino & Espinha (2005a, b). Ela fornece uma representação *compacta* (em relação ao custo de memória) para malhas com fronteira externa com topologia manifold (Mäntylä, 1988), uma vez que apenas as entidades topológicas elemento e nó são armazenadas em memória. Ao mesmo tempo, é considerada *completa* (Weiler, 1986, 1988), no sentido em que permite que todas as relações de adjacências entre entidades

topológicas da malha sejam obtidas em ordem proporcional ao número de entidades retornadas. Entre as funcionalidades oferecidas por TopS, de especial interessse a esta pesquisa, encontra-se uma interface uniforme para a representação e tratamento de elementos coesivos. É fornecido suporte sequencial para a inserção adaptativa de elementos coesivos (requerida por simulações de fraturas e fragmentação extrínsecas), a partir da classificação topológica proposta por Paulino et al. (2008).

A representação de malha distribuída proposta neste trabalho utiliza como base a estrutura de dados topológica sequencial TopS (Celes, Paulino & Espinha, 2005a, b). Dessa forma, os principais conceitos de TopS relacionados a este trabalho são revistos a seguir.

2.6.1. Entidades topológicas

Embora diversos tipos de entidades topológicas sejam definidos por TopS, apenas as entidades *elemento* e *nó* são explicitamente representadas. Isso significa que essas entidades possuem uma representação concreta, residindo efetivamente no espaço de memória utilizado pela estrutura de dados. A entidade *elemento* representa qualquer tipo de elemento finito que possa ser definido por um conjunto ordenado (*template*) de nós, o que inclui os principais tipos empregados em análises de elementos finitos (ex. triângulos, tetraedros e hexahedros, de ordem linear, quadrática ou superior) (Bathe, 1996). Alguns dos tipos de elementos, lineares e quadráticos, suportados por TopS são ilustrados na Figura 5. A entidade *nó* representa nós de elementos finitos, podendo estar associados aos cantos (Figura 5a) ou outras posições da fronteira ou interior de um elemento, no caso de elementos não-lineares (ex. elementos quadráticos – Figura 5b). Nós localizados sobre uma aresta e que não correspondem aos cantos da aresta são chamados *nós de meio de aresta* (ou *mid-side nodes*).

Além de elementos e nós, a estrutura de dados também armazena algumas informações de adjacência entre essas entidades, de forma a permitir o acesso eficiente a elas. Assim, além da referência tradicional – conectividade nodal – de um elemento ao seu conjunto de nós (relação *elemento-a-nó*), um elemento possui uma referência para cada elemento adjacente a ele (relação *elemento-a-elemento*). Da mesma forma, todo nó possui uma referência para um de seus elementos adjacentes (relação *nó-a-elemento*), considerando-se uma

malha regular de domínio manifold. Todas as outras relações de adjacência fornecidas por TopS são derivadas a partir das relações anteriores.

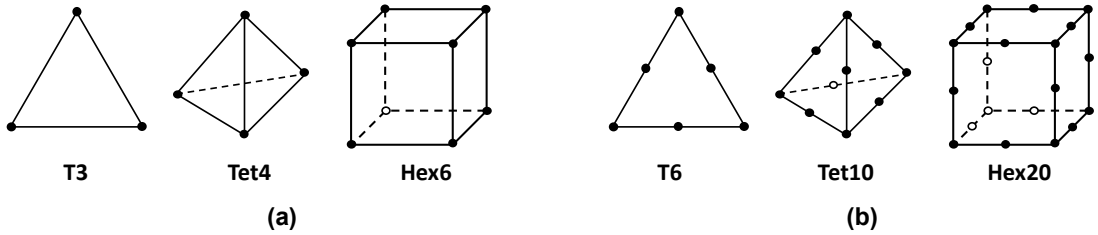

Figura 5 - Exemplos de elementos volumétricos (a) lineares e (b) quadráticos suportados pela estrutura topológica TopS.

As entidades topológicas *vértice*, *aresta*, *face* e *faceta* são representadas implicitamente pela estrutura de dados. Dessa forma, são criadas sob demanda, de forma transparente, sempre que requisitadas pela aplicação cliente. Um *vértice* representa a entidade de dimensão 0 associada a um nó de canto de um elemento finito; *aresta* é definida como a entidade de dimensão 1 limitada por dois vértices; e *face* é a entidade de dimensão 2 limitada por um conjunto de arestas. A entidade *faceta* é definida por TopS para representar a interface entre dois elementos volumétricos adjacentes, ou entre um elemento e a fronteira da malha. Em uma malha bidimensional, uma faceta equivale à entidade de dimensão 1 (aresta), enquanto que, em malhas tridimensionais, equivale à entidade de dimensão 2 (face). Com isso, a entidade faceta provê uma abstração conveniente para o tratamento uniforme de operações que atuam nas interfaces entre elementos, como a inserção de elementos coesivos.

Além de *vértice*, *aresta*, *face* e *faceta*, TopS define um conjunto de entidades implícitas adicionais, que representam os *usos* daquelas entidades pelos elementos adjacentes: *vértice-uso* (*vertex-use*), *aresta-uso* (*edge-use*), *face-uso* (*face-use*) e *faceta-uso* (*facet-use*). Os usos de uma faceta compartilhada por dois elementos são ilustrados na Figura 6. Uma *faceta-uso* independente é definida para cada elemento que compartilha (ou usa) a faceta. Vértices e arestas podem ser compartilhados por um número arbitrário de elementos, e assim são definidos vários usos de uma mesma entidade.

Cada elemento isoladamente é limitado por um conjunto de faces, arestas e vértices, mapeados para os correspondentes usos de entidades do elemento. Uma vez que a topologia local do elemento, em geral, depende unicamente do seu tipo (ex. Tet4 (tetraedro linear), Brick20 (hexaedro quadrático), etc.), pode-se

definir uma *ordenação topológica local fixa* (ou *element template*) para as referências às entidades locais do elemento. A ordenação fornece acesso direto a todas as relações de adjacência entre entidades no interior dos elementos, sendo reutilizada por todos os elementos de um mesmo tipo.

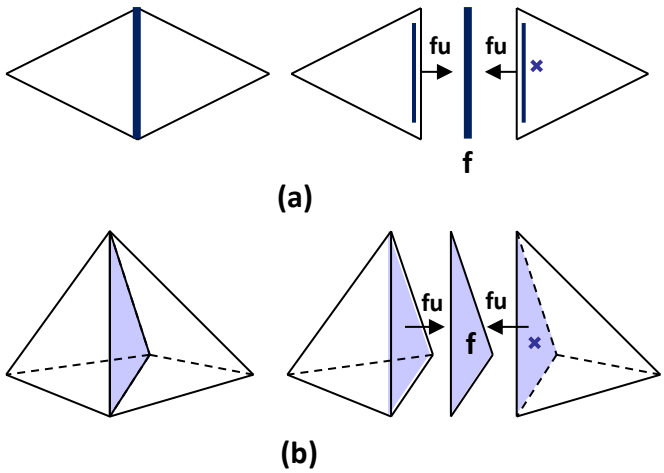

Figura 6 - A faceta *f* é *usada* pelos dois elementos adjacentes a ela, em 2D (a) ou em 3D (b). Para cada elemento que usa a faceta, define-se uma *faceta-uso* (*fu*) correspondente. A faceta é univocamente representada por um de seus usos, no qual diz-se que ela está *ancorada*. O uso que representa a faceta é destacado pela marca `x`.

Em TopS, entidades são acessadas por meio de *referências opacas* (*opaque handles*) retornadas pela estrutura de dados à aplicação cliente no lugar da própria entidade. A utilização de referências opacas fornece à aplicação cliente uma interface transparente para o tratamento uniforme tanto de entidades explícitas quanto implícitas. Uma referência opaca consiste em: um identificador de elemento (E_i) ou de nó (N_i), ou o par (E_i, id_{local}), no caso de entidades implícitas. Um uso de entidade implícita (vértice-uso, aresta-uso, face-uso ou faceta-uso) é unicamente representado por (E_i, id_{local}), que é composto por uma referência ao elemento (E_i) ao qual ele está associado, mais o índice da entidade (id_{local}) em relação à ordenação topológica fixa do elemento.

As entidades implícitas vértice, aresta, face e faceta são representadas por um de seus usos pelos elementos adjacentes. Com isso, duas arestas que possuem os mesmos nós podem ser identificadas pela estrutura de dados como arestas diferentes. Esse tipo de configuração, que é comum em elementos coesivos (Seção 2.6.3), não poderia ser representado se as arestas fossem definidas através de outras entidades de ordem inferior (ex.: nós).

De forma a identificar o uso de entidade que representa uma entidade topológica do tipo vértice, aresta, face ou faceta, define-se um marcador binário, chamado *âncora*, que é colocado no elemento adjacente correspondente ao uso que representa a entidade. Isso é mostrado na Figura 6. Embora dois elementos compartilhem uma mesma faceta, apenas um deles possui o marcador de âncora da entidade (ilustrado pela marca 'x' na figura). Assim, diz-se que a faceta encontra-se *ancorada* naquele elemento, e a faceta-uso correspondente é considerada a *representante* da faceta. Em TopS, a representação de entidades implícitas é sempre associada aos elementos adjacentes a elas.

A Figura 7 ilustra a visitação de todos os usos de um vértice pelos elementos adjacentes, em uma malha bidimensional. Um procedimento equivalente é realizado para arestas e malhas tridimensionais (Celes, Paulino & Espinha, 2005a). A partir de um vértice (*v*), obtém-se o vértice-uso que o representa (*vu0*), a partir do elemento adjacente contendo o marcador de âncora correspondente. Usando-se a ordenação local fixa do elemento, acessa-se uma de suas facetas-uso (*fu0*) adjacentes ao vértice-uso (*vu0*). Em seguida, com base na relação de adjacência elemento-a-elemento armazenada pela estrutura de dados, obtém-se a faceta-uso (*fu1*) correspondente no elemento adjacente, e, novamente usando-se a ordenação local fixa do elemento, obtém-se o vértice-uso (*vu1*) associado ao vértice (*v*). O procedimento continua em cada elemento até que todos os usos do vértice tenham sido visitados.

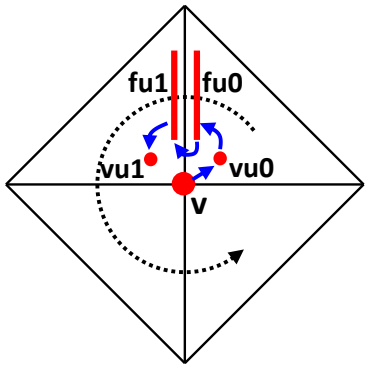

Figura 7 - Visitação dos usos de um vértice pelos elementos adjacentes em uma malha bidimensional.

2.6.2. Elementos coesivos

Do ponto de vista da representação topológica, elementos coesivos (Pandolfi & Ortiz, 1998, 2002; Paulino et al., 2008) são tipos de elementos especiais que consistem de apenas duas facetas. Esses elementos são criados

nas interfaces entre dois elementos volumétricos adjacentes para representar a ocorrência de fraturas na malha de elementos finitos. A Figura 8 mostra dois elementos coesivos (CohE3 e CohT6) compatíveis com facetas de dois elementos volumétricos (T6 e Tet10).

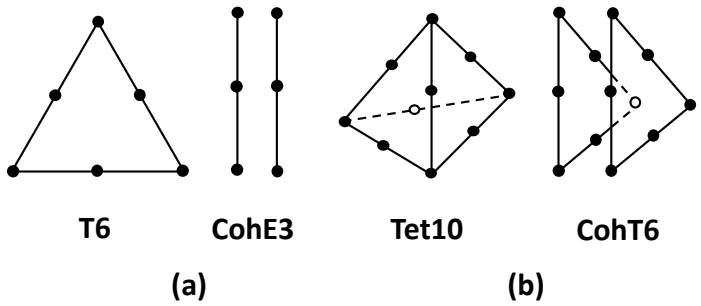

Figura 8 - Exemplos de elementos volumétricos 2D (a) e 3D (b) e elementos coesivos correspondentes (CohE3 e CohT6).

Elementos coesivos podem ter nós compartilhados entre as suas duas facetas, *f1* e *f2* . Dessa forma, a seguinte incidência nodal é válida para um elemento do tipo Coh2E3: *f1*:(n_A, n_B, n_C), *f2*:(n_A, n_D, n_C), onde os primeiros três nós correspondem à primeira faceta do elemento (*f1*) e os restantes à segunda (*f2*). Apesar do compartilhamento de nós entre as facetas, diferentes vértices-usos, arestas-usos e faces-usos são definidos para cada faceta do elemento coesivo. Alguns exemplos de possíveis incidências nodais de elementos coesivos são mostrados na Figura 9.

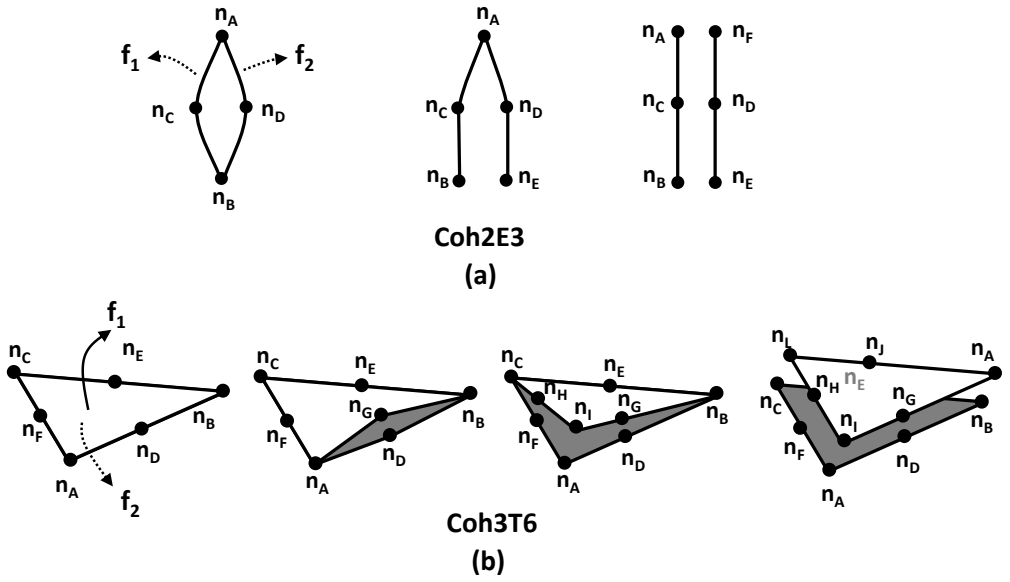

Figura 9 - Exemplos de possíveis incidências nodais de elementos coesivos: (a) 2D (Coh2E3); e (b) 3D (Coh3T6). Elementos coesivos podem possuir nós compartilhados pelas duas facetas diferentes.

Da mesma forma que outros tipos de elementos (volumétricos), TopS representa elementos coesivos explicitamente e os trata como elementos regulares da malha de elementos finitos. Assim, elementos coesivos também possuem uma ordenação topológica (*template*) local fixa e podem ter atributos associados. Isso difere de outras abordagens, em que elementos coesivos são tratados como atributos de elementos volumétricos (Pandolfi & Ortiz, 1998, 2002).

2.6.3. Inserção adaptativa sequencial de elementos coesivos

A estrutura de dados topológica sequencial TopS permite que elementos coesivos sejam inseridos adaptativamente, conforme necessário. A inserção adaptativa baseia-se na classificação sistemática de facetas proposta por Paulino et al. (2008). Essa classificação fornece uma maneira consistente de se identificarem as operações topológicas necessárias à atualização da estrutura de dados, após a introdução de um novo elemento coesivo. Uma vantagem é que ela pode ser empregada uniformemente para qualquer tipo de elemento, tanto em malhas bidimensionais (2D) quanto tridimensionais (3D).

Os passos para a inserção de um elemento coesivo ao longo da faceta entre os elementos *E1* e *E2* são ilustrados na Figura 10 e brevemente descritos a seguir:

1. Cria-se um elemento coesivo entre *E1* e *E2*. As adjacências dos elementos *E1* e *E2* são atualizadas, de forma que *E1* e *E2* não são mais adjacentes um ao outro, mas ao elemento coesivo (Figura 10a e Figura 10d). Inicialmente, as duas facetas do elemento coesivo possuem o mesmo conjunto de nós.
2. Para cada aresta-uso (*eu*) da faceta de *E1* compartilhada com o elemento coesivo (Figura 10b e Figura 10e), visitam-se todos os outros usos da aresta correspondente, considerando as adjacências atualizadas no passo anterior. Se o elemento *E2* não foi alcançado, duplica-se a aresta. Os nós de meio de aresta, se existentes, também devem ser duplicados.
3. O procedimento de vértices é similar ao de arestas. Para cada vértice-uso (*vu*) da faceta de *E1* compartilhada com o elemento coesivo (Figura 10c e Figura 10f), visitam-se todos os outros usos

do vértice correspondente, considerando-se as adjacências atualizadas. Se *E2* não foi alcançado, duplica-se o vértice e o nó de canto correspondente.

As conectividades de todos os elementos envolvidos precisam ser atualizadas para cada nó duplicado. Isso é feito pela substituição do nó original pelo novo nos elementos visitados durante os passos 2 e 3. Elementos não visitados não são alterados. Casos correspondentes à inserção de elementos coesivos em uma malha bidimensional são ilustrados na Figura 11.

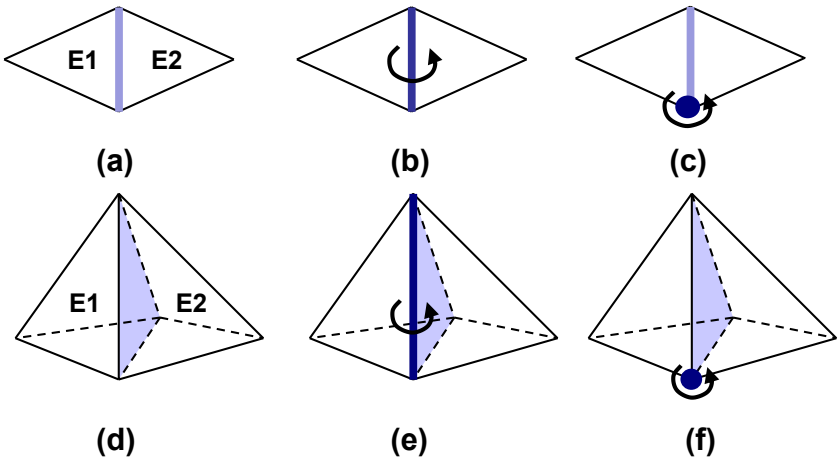

Figura 10 - Operações para a atualização da topologia da malha quando um novo elemento coesivo é inserido, em 2D (a - c) e em 3D (d - f). (a), (d) - O elemento coesivo é criado na faceta entre *E1* e *E2*. (b), (e) - Para cada aresta da faceta inicial, a partir de *E1*, os usos correspondentes são visitados. Se o uso de *E2* não é alcançado, a aresta e nós de meio da aresta são duplicados. (c), (f) - Para cada vértice da faceta inicial, a partir de *E1*, os usos correspondentes são visitados. Se o uso de *E2* não é alcançado, o vértice e o nó correspondente são duplicados.

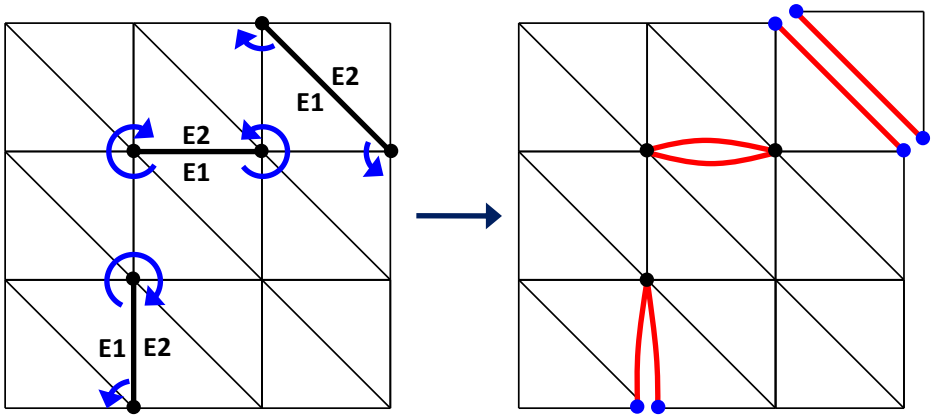

Figura 11 - Casos correspondentes à inserção de elementos coesivos em uma malha bidimensional (Paulino et al., 2008).

2.6.4. Conjuntos de atributos *densos* e *esparsos*

Para realizar uma simulação numérica, a aplicação cliente pode precisar associar atributos próprios da simulação a entidades topológicas da malha de elementos finitos. A fim de facilitar a gerência de atributos por aplicações que utilizam malhas dinâmicas é fornecido suporte para *conjuntos de atributos,* alocados automaticamente por TopS. Dois tipos são definidos: *atributos densos* (*dense attributes*) e *esparsos* (*sparse attributes*). Conjuntos *densos* são otimizados para dados associados a todas (ou quase todas) as entidades de um dado tipo (ex. dados da simulação numérica em cada nó ou elemento), enquanto que conjuntos *esparsos* são tipicamente empregados na representação de dados associados a apenas parte das entidades (ex. condições de contorno, ou forças aplicadas).

Os conjuntos de atributos *densos* se aplicam apenas a entidades explícitas (nós e elementos), sendo um conjunto diferente criado para cada tipo de elemento ou nó. A implementação consiste em um vetor dinâmico de dados, com número de posições igual ao de entidades do tipo ao qual o vetor está associado. Os índices do vetor de atributos correspondem aos mesmos índices no vetor da estrutura de dados contendo os nós ou elementos relacionados (TopS utiliza vetores distintos para representar cada tipo de elemento ou nó). O tamanho alocado para cada valor de atributo (posição do vetor) é fixo e definido pela aplicação durante a criação do conjunto. O vetor dinâmico é redimensionado automaticamente, com base no número de entidades representadas. Vários conjuntos de atributos podem ser criados para o mesmo

tipo de entidade, sendo que cada conjunto associado a um tipo de entidade possui um identificador único, usado pela aplicação para acessá-lo.

Ao contrário de atributos densos, conjuntos de atributos *esparsos* podem ser criados para quaisquer das entidades representadas por TopS, explícitas ou implícitas. Assim como os atributos densos, os esparsos também são definidos para cada tipo de elemento ou nó ou de entidade implícita. A implementação consiste em uma tabela associativa dinâmica (ou uma *tabela de dispersão* – *hash table*) indexada pelos identificadores das entidades correspondentes. A criação de conjuntos esparsos é feita da mesma maneira que os densos.