Universidade Federal de Pernambuco – UFPE

Centro de Tecnologia e Geociências

Programa de Pós-Graduação em Engenharia Civil

Diogo Tenório Cintra

Metodologia de paralelização híbrida do DEM com controle de balanço de carga baseado em curva de Hilbert

Recife/2016

Diogo Tenório Cintra

Metodologia de paralelização híbrida do DEM com controle de balanço de carga baseado em curva de Hilbert

Tese apresentada ao Programa de Pós-Graduação em Engenharia Civil como parte dos requisitos para a obtenção do título de Doutor em Engenharia Civil.

Orientador: Prof. Ramiro Brito Willmersdorf, PhD

Coorientador: Prof. Paulo Roberto Maciel Lyra, PhD

Recife/2016

Catalogação na fonte
Bibliotecária Valdicéa Alves, CRB-4 / 1260

C574m Cintra, Diogo Tenório.
Metodologia de paralelização híbrida do DEM com controle de balanço de carga baseado em curva de Hilbert / Diogo Tenório Cintra. - 2016
181folhas, Il. Abr.Sigl e Tabs.

Orientador: Profº. Dr. Ramiro Brito Willmersdorf.
Coorientador: Profº Dr. Paulo Roberto Maciel Lyra.

Tese (Doutorado) – Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação Engenharia Civil, 2016.

Inclui Referências.

1. Engenharia Civil. 2. Método dos elementos discretos.
3. Processamento de alto desempenho. 4. Paralelização híbrida. 5. HSFC.
I. Willmersdorf, Ramiro Brito. (Orientador). II. Lyra, Paulo Roberto Maciel. (Coorientador). III. Título.

UFPE

624 CDD (22. ed.) BCTG/2016-121

UNIVERSIDADE FEDERAL DE PERNAMBUCO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA CIVIL

A comissão examinadora da Defesa de Tese de Doutorado

METODOLOGIA DE PARALELIZAÇÃO HÍBRIDA DO DEM COM CONTROLE DE BALANÇO DE CARGA BASEADO EM CURVA DE HILBERT

defendida por

Diogo Tenório Cintra

Considera o candidato APROVADO

Recife, 29 de janeiro de 2016

Orientadores:

_____
Prof. Dr. Ramiro Brito Willmersdorf – UFPE
(orientador)

_____
Prof. Dr. Paulo Roberto Maciel Lyra – UFPE
(coorientador)

Banca Examinadora:

_____
Prof. Dr. Ramiro Brito Willmersdorf – UFPE
(orientador)

_____
Prof. Dr. Manoel Eusébio de Lima – UFPE
(examinador externo)

_____
Prof. Dr. José Maria Andrade Barbosa – UFPE
(examinador externo)

_____
Prof. Dr. Ézio da Rocha Araújo – UFPE
(examinador externo)

_____
Prof. Dr. Darlan Karlo Elisiário de Carvalho – UFPE
(examinador interno)

Aos meus pais, pela atenção, apoio e amor incondicional.

Agradecimentos

A Deus, por tornar tudo possível.

Aos meus pais e ao meu irmão Felipe, por todo incentivo que motivou o desenvolvimento deste trabalho.

À minha noiva, Shirley Viana, por todo o apoio, carinho e paciência demonstrados durante todos os dias, horas e minutos em que estive imerso no universo das partículas e distante do mundo concreto.

Aos meus colegas e amigos de trabalho no LCCV, em especial ao Fábio Ferreira e à Luciana Vieira, pelo apoio em tarefas do dia a dia, dúvidas e sugestões.

Aos demais amigos e familiares que contribuiram de forma direta ou indireta na superação deste desafio.

Aos professores Paulo Roberto Maciel Lyra e Ramiro Brito Willmersdorf, pelas orientações, conselhos e incentivos.

Aos professores Márcio Muniz de Farias, José Maria Andrade Barbosa, Manoel Eusébio de Lima, Ézio da Rocha Araújo e Darlan Karlo Elisiário de Carvalho pelas sugestões de melhorias apontadas no exame de qualificação e na defesa desta tese.

Ao Programa de Pós-Graduação em Engenharia Civil (UFPE), docentes e funcionários, pela qualidade e empenho na formação acadêmica.

Ao Laboratório de Computação Científica e Visualização - LCCV/UFAL, pela infraestrutura de apoio e incentivo à pesquisa e desenvolvimento computacional, em especial aos professores Adeildo Ramos, Eduardo Setton, Eduardo Nobre e William Wagner.

"Os computadores são incrivelmente rápidos, precisos e burros;
os homens são incrivelmente lentos, imprecisos e brilhantes;
juntos, seus poderes ultrapassam os limites da imaginação."

Albert Einstein

Resumo

Esta tese apresenta uma metodologia de paralelização híbrida aplicada ao Método dos Elementos Discretos (DEM - *Discrete Element Method*) que combina MPI e OpenMP com o intuito de melhoria de desempenho computacional. A metodologia utiliza estratégias de decomposição de domínio visando a distribuição do cálculo de modelos de larga escala em um *cluster*. A técnica proposta também particiona a carga de trabalho de cada subdomínio entre *threads*. Este procedimento adicional visa obter maiores desempenhos computacionais através do ajuste de utilização de mecanismos de troca de mensagens entre processos e paralelização por *threads*. O objetivo principal da técnica é reduzir os elevados tempos de comunicação entre processos em ambientes computacionais de memória compartilhada tais como os processadores modernos. A divisão de trabalho por *threads* emprega a curva de preenchimento de espaço de Hilbert (HSFC) visando a melhoria de localidade dos dados e evitando custos computacionais (*overheads*) resultantes de ordenações constantes para o vetor de partículas. As simulações numéricas apresentadas permitem avaliar os métodos de decomposição de domínio, técnicas de particionamento, mecanismos de controle de acesso à memória, dentre outros. Algoritmos distintos de particionamento e diferentes estratégias de solução paralela são abordados para ambientes computacionais de memória distribuída, compartilhada ou para um modelo híbrido que envolve os dois ambientes. A metodologia desenvolvida e a ferramenta computacional utilizada nas implementações realizadas, o *software* DEMOOP, fornecem recursos que podem ser aplicados em diversos problemas de engenharia envolvendo modelos de partículas em larga escala. Nesta tese alguns destes problemas são abordados, em especial aqueles relacionados com fluxo de partículas em rampas, em funis de descarga e em cenários reais de deslizamento de terra. Os resultados mostram que as estratégias de execução híbridas atingem, em geral, melhores desempenhos computacionais que aqueles que se baseiam unicamente em troca de mensagens. A técnica de paralelização híbrida desenvolvida também obtém um bom controle de balanço de carga entre *threads*. Os estudos de caso apresentados apresentam boa escalabilidade e eficiências paralelas. O método proposto permite uma execução configurável de modelos numéricos do DEM e introduz uma estratégia combinada que melhora localidade dos dados e um balanceamento de carga iterativo.

Palavras-chaves: Método dos elementos discretos. Processamento de alto desempenho. Paralelização híbrida. HSFC.

Abstract

This thesis introduces a methodology of hybrid parallelization applied to the Discrete Element Method (DEM) that combines MPI and OpenMP to improve computational performance. The methodology uses domain decomposition strategies to distribute the computation of large-scale models in a cluster. It also partitions the workload of each subdomain among threads. This additional procedure aims to reach higher computational performance by adjusting the usage of message passing artifacts and threads. The main objective is to reduce the expensive communications between processes in computer resources of shared memory such as modern processors. The work division by threads employs Hilbert Space Filling Curves (HSFC) in order to improve data-locality and to avoid the overhead caused by the dynamical sorting of the particles array. Presented numerical simulations allow to evaluate several domain decomposition schemes, partitioning methods, mechanisms of memory access control, among others. The work investigate distinct schemes of parallel solution for both distributed and shared memory environments. The method and the computational tool employed, the software DEMOOP, provide applied resources for several engineering problems involving large scale particle models. Some of these problems are presented on this thesis, such as the particle flows that happen on inclined ramps, discharge hoppers and real scenarios of landslides. The results shows that the hybrid executions reach better computational performance than those based on message passing only, including a good control of load balancing among threads. Case studies present good scalability and parallel efficiencies. The proposed approach allows a configurable execution of numerical models and introduces a combined scheme that improves data-locality and an iterative workload balancing.

Key-words: DEM. High performance computing. Hybrid parallelization. HSFC.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BCC	*Body Centered Cubic*
CPU	*Central Processing Unit*
CUDA	*Compute Unified Device Architecture*
DEM	*Discrete Element Method*
DEMOOP	*Discrete Element Method Object Oriented Programming*
DEMPAD	*Discrete Element Method* com Programação de Alto Desempenho
DDA	*Discontinua Deformation Analysis*
FCC	*Face Centered Cubic*
GPU	*Graphics Processing Unit*
HCP	*Hexagonal Closed-Packed*
HSFC	*Hilbert Space-Filling Curve*
K-L	*Kernighan-Lin*
LCCV	Laboratório de Computação Científica e Visualização
MPI	*Message Passing Interface*
NBS	*No Binary Search*
PetroDEM	*Petrobras Discrete Element Method*
PMRSB	*Parallel Multilevel Recursive Spectral Bisection*
PPM	*Parallel Particle-Mesh*
RCB	*Recursive Coordinate Bisection*
RGB	*Recursive Graph Bisection*
RIB	*Recursive Inertial Bisection*
RSB	*Recursive Spectral Bisection*
SC	*Simple Cubic*

URB	*Unbalanced Recursive Bisection*
VLSI	*Very-Large-Scale Integration*
XDMF	*eXtensible Data Model and Format*

Sumário

1 Introdução

1.1 Motivação

Na análise de alguns problemas da engenharia, sobretudo aqueles da geomecânica, é comum o estudo do comportamento mecânico de meios de natureza descontínua. Isso ocorre, por exemplo, na análise de problemas que envolvem meios granulares, tais como: estudos de estabilidade de taludes, deslizamento de terra e avalanches (CAMPBELL; CLEARY; HOPKINS, 1995; CHANG; TABOADA, 2009; WALTHER; SBALZARINI, 2009), problemas de interação solo-estrutura, cravação e perfuração (ONATE; ROJEK, 2004), fraturamento (OWEN et al., 2004), contato e impacto (BROWN et al., 2000) e em estudos de problemas da geologia estrutural ou geofísica (ABE; GENT; URAI, 2011) (*Sandbox*). Um procedimento eficaz para o estudo de problemas desta natureza é apresentado por Cundall e Strack (1979). O método proposto por Cundall e Strack (1979) investiga o fluxo de matéria granular através de elementos discretos. Os elementos discretos representam corpos dotados de massa e forma geométrica qualquer, livres para movimentar-se no espaço e que interagem entre si através de contatos ou forças coesivas. O denominado Método dos Elementos Discretos (DEM - *Discrete Element Method*) faz uso de entidades com estas características para representar meios descontínuos diversos. Por simplificação, os elementos discretos são, normalmente, representados por partículas, denominação dada aos corpos com geometrias circulares ou esféricas. A formulação lagrangeana empregada é baseada na mecânica Newtoniana e o modelo *soft-sphere* é muito empregado para representar o fenômeno de contato.

Em aplicações práticas do DEM é comum a utilização de uma quantidade muito grande de partículas para representar com maior fidelidade um determinado domínio de análise. Problemas de larga escala, a exemplo do apresentado por Walther e Sbalzarini (2009), podem envolver meios contendo centenas de milhões de partículas. No caso particular da aplicação do trabalho de Walther e Sbalzarini, avalia-se o comportamento dinâmico de uma avalanche de areia em um plano inclinado fazendo o uso de cerca de 122 milhões de partículas. Uma vez que o cálculo da dinâmica das partículas é realizado de maneira quase independente, o aumento do número de partículas do modelo resulta, portanto, no aumento da demanda por recursos computacionais e consequentemente no tempo necessário para a simulação numérica. Algumas estratégias de solução, no entanto, permitem que grupos de partículas sejam representados em conjuntos cujas dinâmicas são determinadas de forma mais grosseira (RYCROFT et al., 2006; SAKAI et al., 2010). Porém, estas técnicas ainda continuam, mesmo que em menor relação, dependentes do número de partículas. Outra alternativa para aumentar o desempenho da simulação de elementos discretos é permitir

que o avanço da solução ao longo do tempo ocorra de forma mais rápida (ALEXIADES; AMIEZ; GREMAUD, 1996) ou mesmo adaptativa (CINTRA et al., 2008; LAGES et al., 2013). Estas técnicas devem considerar as elevadas frequências naturais do sistema para que o processo de integração numérica temporal não acarrete erros significativos.

Sistemas computacionais que implementem o DEM devem, portanto, estar preparados para manipular, de forma otimizada, uma grande necessidade de processamento e dados. Um dos grandes desafios é o tempo de processamento que as simulações requerem. Visando a atenuação deste problema, algumas técnicas de paralelização de processamento têm sido utilizadas. Podem-se citar aquelas que envolvem processadores numéricos ou gráficos, *CPU's* e *GPU's*, em ambientes de memória compartilhada ou distribuída. Para tanto, dentre outros, utilizam-se recursos de programação paralela tais como os padrões *OpenMP* (CHAPMAN; JOST; PAS, 2007) e *MPI* (HUGHES; HUGHES, 2003), e a arquitetura *CUDA* (NGUYEN, 2007) para a programação em *GPU* (OWENS et al., 2007).

As técnicas de paralelização envolvendo ambientes de memória compartilhada, quando aplicadas ao algoritmo do DEM, são concebidas de forma a evitar condições de concorrência de acesso à memória (processos competindo por acesso simultâneo à memória para leitura e gravação de dados) na determinação das forças atuantes nas partículas (NISHIURA; SAKAGUCHI, 2011). Este problema ocorre devido ao mecanismo de cálculo das forças de contato, quando mais de um processo disputa pelo acesso à memória correspondente a uma partícula. Além disto, deve-se promover um balanceamento de carga eficaz que considere uma definição equilibrada de pares de contato (NISHIURA; SAKAGUCHI, 2011) para os processos, bem como técnicas para a redução do uso de memória (SHIGETO; SAKAI, 2011).

Embora as estratégias de paralelização em memória compartilhada (*OpenMP*) sejam de rápida implementação e exijam pouco conhecimento do código-fonte, as estratégias de decomposição de domínio são mais aplicáveis em sistemas de maior porte. Geralmente estas estratégias são empregadas em ambientes de memória distribuída onde a comunicação entre processos é feita através de troca de mensagens (ABE; PLACE; MORA, 2004; WANG et al., 2006; MAKNICKAS et al., 2006; CHANG; TABOADA, 2009). Alguns sistemas computacionais consideram, além da paralelização de código, decomposições adaptativas de domínio, balanceamento de carga e entrada e saída de dados (*IO*) em paralelo (SBALZARINI et al., 2006a; WALTHER; SBALZARINI, 2009).

As *GPUs* apresentam-se ao longo dos anos como alternativas de processamento bastante aplicáveis no estudo de meios particulados. Seu desenvolvimento em termos de *hardware* tem sido aplicado para atender a demanda tecnológica da indústria de diversos setores (CHEN et al., 2009; RADEKE; GLASSER; KHINAST, 2010; XU et al., 2011). Supercomputadores, alguns da ordem de 1 *Petaflops* de pico de performance (CHEN et al., 2009), são utilizados com o intuito de atender as dificuldades tecnológicas enfrentadas.

Diversos simuladores numéricos que fazem uso de arquiteturas envolvendo *GPUs* são desenvolvidos portanto no intuito de tornar mais rápida a simulação de modelos numéricos de larga escala.

Em análises de grande porte envolvendo o DEM, o uso de técnicas de computação de alto desempenho é imprescindível. Este emprego pode viabilizar análises mais sofisticadas dos meios descontínuos envolvidos. Isso ocorre tanto devido à distribuição de tarefas por processador, que reflete na diminuição do tempo de processamento da simulação, como também, em alguns casos, devido à distribuição de memória de trabalho para alocação das estruturas de dados manipuladas. Diversas questões envolvendo técnicas de paralelização do algoritmo do DEM ainda despertam interesse da comunidade científica. Um dos grandes desafios consiste em aumentar a eficiência destas técnicas em *clusters* cada vez maiores, envolvendo centenas ou milhares de núcleos de processamento.

Deve-se ter em mente ainda que as técnicas de computação de alto desempenho englobam outras atividades não puramente de paralelização, a exemplo de otimizações de código, desenvolvimento de novos algoritmos, dentre outros. Um tópico não muito explorado em códigos científicos diz respeito ao uso otimizado da memória *cache* de processamento. Em geral, quando o uso deste recurso computacional é feito de forma irregular a velocidade aparente de processamento é prejudicada (BADER, 2012). Isto acontece frequentemente quando durante a simulação não existe uma boa correlação entre localidade de dados e localidade espacial.

1.2 Estado da Arte

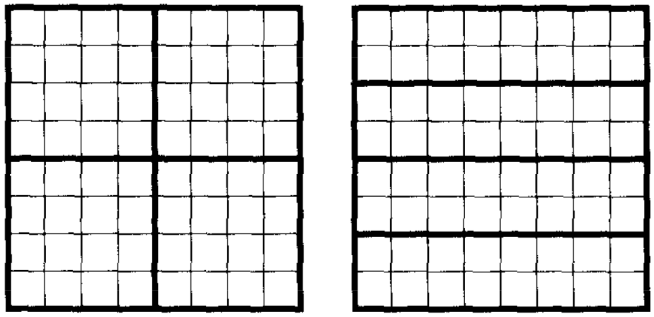
O uso massivo de computadores paralelos para a simulação de partículas, já vem sendo registrado desde antes do inicio da década de 1990. Na época, era bastante comum o uso de computadores vetoriais (RAPAPORT, 1988; GREST; DUNWEG; KREMER, 1989; FORM; ITO; KOHRING, 1993) para a simulação de sistemas contendo vários milhões de partículas (FORM; ITO; KOHRING, 1993). O interesse científico na área era compartilhado no estudo de problemas de diversas naturezas, que tipicamente podem ser classificados de acordo com a forma de interação entre as partículas.

Basicamente, as interações entre partículas podem ser de longo ou curto alcance. As interações de longo alcance são aquelas que atuam a distância, a exemplo das forças eletromagnética e gravitacional. Já as interações de curto alcance, são aquelas que surgem das interações de impacto ou contato, quando existe ou não o choque entre elementos respectivamente. O DEM baseia-se no cálculo de forças decorrentes de contato, e portanto classifica-se como um método com interações de curto alcance.

As técnicas de paralelização de código aplicadas em sistemas onde a carga varia durante a execução são compostas por duas etapas básicas: a de decomposição (particiona-

mento) de domínio e a de balanceamento dinâmico de carga. Na etapa de decomposição, divide-se o domínio total de simulação em subdomínios de menor dimensão. Os subdomínios são, desta forma, distribuídos entre os processos. Em termos práticos de simulações de partículas, isso consiste em dividir o cálculo do movimento das partículas do modelo entre os processos paralelos. Dentre as técnicas mais comuns de particionamento bidimensional estão aquelas que se baseiam na divisão geométrica do domínio em faixas ou em células, conforme ilustra a Figura 1.

Figura 1 – Decomposição em quatro subdomínios: blocos (esquerda), faixas (direita).



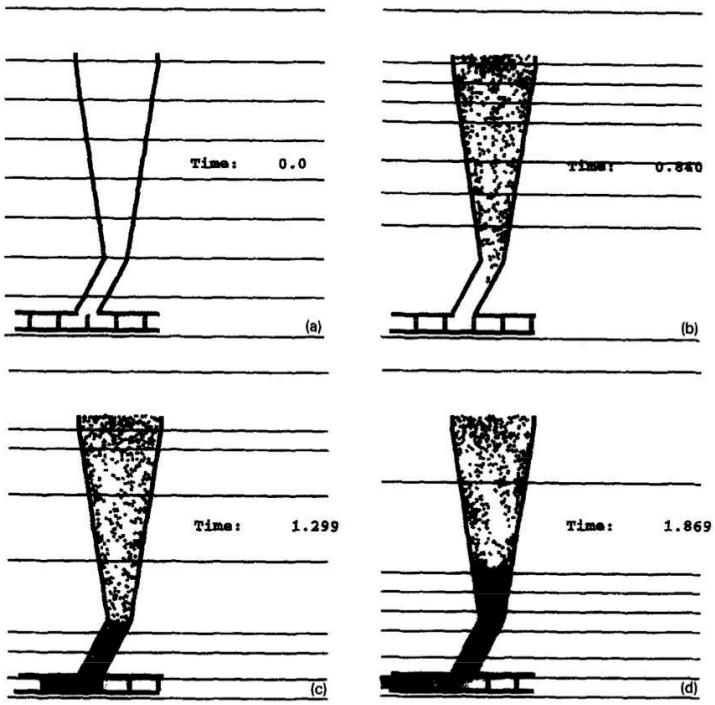
Fonte: Kohring (1995).

A etapa de balanceamento dinâmico de carga consiste em ajustar, ao longo do tempo, o particionamento já obtido para tentar equilibrar o tempo de CPU entre os processos. Na prática isso pode ser atingido movendo partículas de um subdomínio (processo) para outro. Deve-se ter em mente que as partículas são livres para se movimentar no espaço. Ocasionalmente, podem sair do subdomínio de cálculo de um processo e entrar no subdomínio de cálculo do processo "vizinho", no caso de particionamentos geométricos. Esse movimento gera um desbalanceamento de carga, onde um processo leva mais tempo que os demais para o cálculo do movimento das suas partículas, e deve ser devidamente tratado para não comprometer a eficiência da simulação paralela.

No trabalho de Kohring (1995), desenvolve-se uma estratégia de particionamento e balanceamento de carga para interações de curto alcance. A técnica baseia-se em uma variante não linear do método de difusão. Considera-se, por simplicidade, uma decomposição baseada em faixas para a divisão do domínio de análise. Desta forma as faixas delimitam os domínios de cálculo dos processos envolvidos na simulação paralela. A Figura 2 ilustra o método de balanceamento dinâmico de carga, para um problema de escoamento de partículas em um funil.

Note que as espessuras das faixas alteram a medida que a simulação avança ao longo do tempo. Isso é feito de maneira a tentar equalizar o número de partículas em cada faixa, e consequentemente balancear a carga de trabalho dos processos. Cada processo calcula portanto sua própria carga medindo o tempo de CPU desde o último passo de balanceamento de carga. Caso haja seja identificado um desbalanço de tempo de CPU entre

Figura 2 – Balanceamento dinâmico de carga baseado em faixas.



Fonte: Kohring (1995).

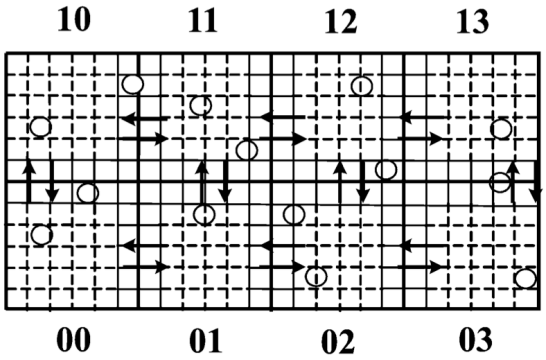
vizinhos, os processo cedem ou recebem conjuntos de células-ligadas (GREST; DUNWEG; KREMER, 1989), alterando as espessuras das faixas e consequentemente a quantidade de partículas nos processos. Devido à natureza discreta da carga, nem sempre é possível obter um estado de perfeito balanço de partículas nos processos.

No trabalho de Maknickas et al. (2006), a estratégia de decomposição de domínio em blocos é empregada na simulação paralela de elementos discretos em ambientes de computação distribuída. Utiliza-se o mesmo conceito de células-ligadas de Grest, Dunweg e Kremer, utilizado por Kohring (1995). A comunicação entre processos, feita na forma de troca de mensagens (MPI), é realizada com o objetivo de receber informações cinemáticas das partículas situadas nas regiões próximas das fronteiras de subdomínios.

Estratégias de particionamento tridimensionais de sistemas de partículas com interação de curto alcance também estão disponíveis na literatura. O mais trivial e comumente utilizado é a decomposição em cubos simples (KACIANAUSKAS et al., 2010). Stijnman, Bisseling e Barkema (2003) apresentam, no entanto, um estudo de eficiência para alguns tipos de formas utilizadas para o particionamento de um domínio tridimensional. Para tanto, utilizam-se células de Voronoi obtidas para as estruturas cristalinas BCC (*Body Centered Cubic*) e FCC (*Face Centered Cubic*), conforme ilustrado na Figura 4, além dos particionamentos em faixas (*slices*) e cúbico simples (*SC*).

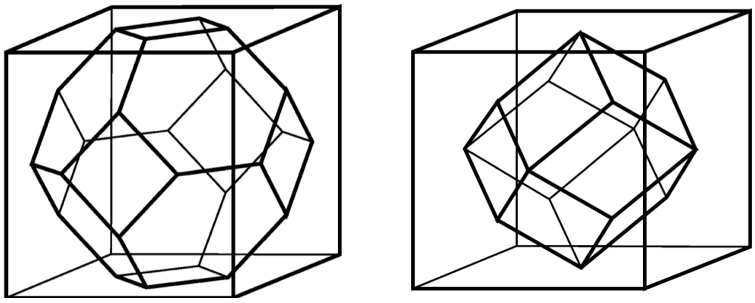
Os dados de eficiência paralela medidos por Stijnman, Bisseling e Barkema (2003), para um cenário de referência (Figura 5), revelam performance aproximadas para os

Figura 3 – Particionamento de domínio em blocos.



Fonte: Maknickas et al. (2006).

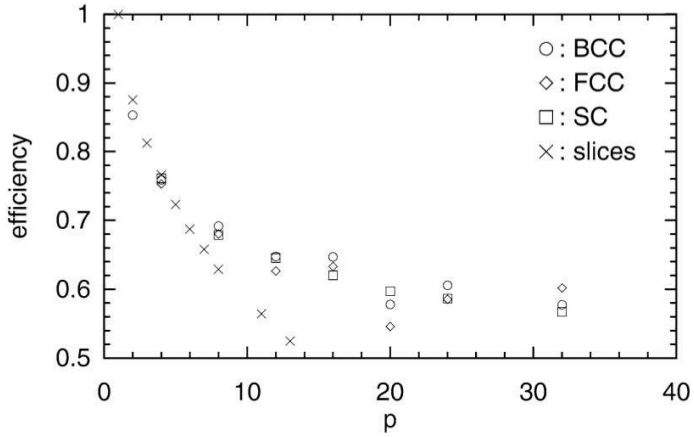
Figura 4 – Células unitárias de Voronoi para arranjos BCC (esquerda) e FCC (direita).



Fonte: Stijnman, Bisseling e Barkema (2003).

arranjos BCC, FCC e cúbico simples. A técnica de particionamento baseada em faixas, no entanto, apresenta o pior resultado para o cenário simulado.

Figura 5 – Eficiência medida como função do número de processos p.

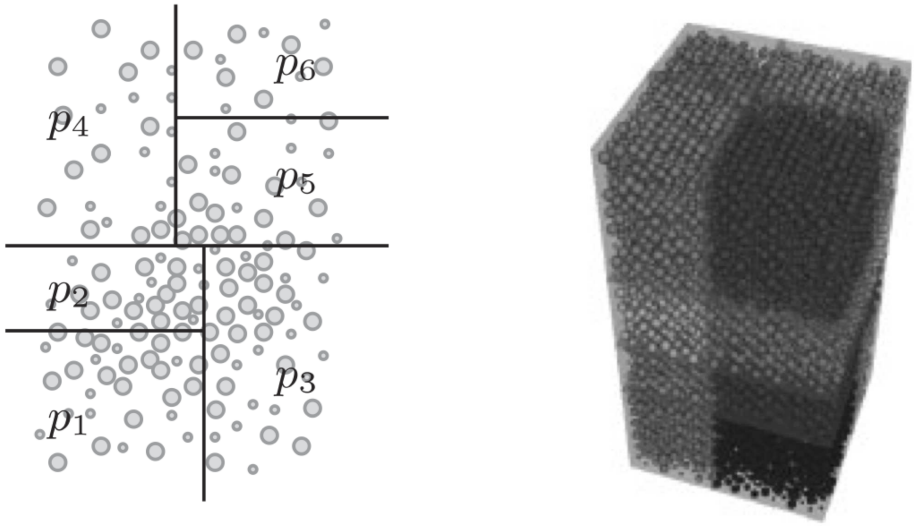


Fonte: Stijnman, Bisseling e Barkema (2003).

Assim como os métodos de particionamento anteriormente apresentados, outras técnicas de particionamento geométrico também são utilizadas. Os particionamentos por Bisseção Recursiva de Coordenadas (RCB - *Recursive Coordinate Bisection*) e por cubóides, por exemplo, são utilizados na biblioteca PPM (*Parallel Particle-Mesh*) (SBALZARINI

et al., 2006a) e por Sbalzarini et al. (2006b). A biblioteca PPM é utilizada por Walther e Sbalzarini (2009) para uma simulação de fluxo granular em larga escala envolvendo 122 milhões de partículas (192 processadores, Cray XT3, eficiência de 40%). Fleissner e Eberhard (2008) também utiliza a técnica decomposição por Bisseção Ortogonal Recursiva (Figura 6).

Figura 6 – Particionamento baseado em Bisseção Ortogonal Recursiva.



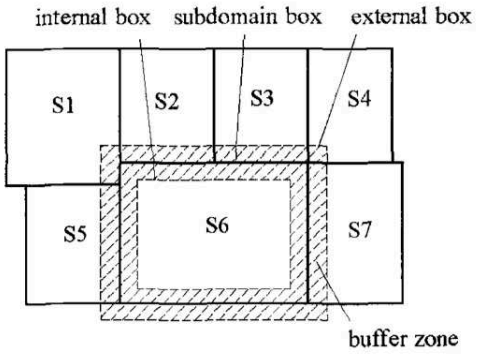
(a) Estratégia de divisão. (b) Particionamento 3D para 16 processos.
Fonte: Fleissner e Eberhard (2008).

A técnica de Bisseção Recursiva de Coordenada (HENDRICKSON; DEVINE, 2000) é utilizada por Wang et al. (2004) na análise paralela de elementos discretos em *clusters*. Esta técnica particiona geometricamente, pela metade, o modelo de acordo com as dimensões mais alongadas. As duas metades são divididas recursivamente então para a distribuição dos subdomínios. A técnica de comunicação entre processos baseia-se em zonas de interface (WANG; FENG; OWEN, 2003), conforme ilustra a Figura 7. A aplicação deste método é bastante comum em simulações de elementos discretos (LUKAS; D'ALBANO; MUNJIZA, 2014). O tamanho das zonas de interface define a frequência de decomposição de domínio, bem como influencia na comunicação entre processos.

Métodos de particionamento e balanceamento de carga baseados em árvores também são utilizados (ZHANG; JIANG; LI, 2009; SHOJAAEE et al., 2012). Na estratégia proposta por Zhang, Jiang e Li, o balanceamento de carga é realizado recursivamente por alongamento e compressão de subdomínios em grupos.

Outra estratégia empregada para o particionamento de domínio é utilizar procedimentos de decomposição em células de Voronoi. Apesar dessa decomposição ser uma forma elegante de particionar subdomínios, possui a desvantagem de necessitar lidar com a natureza complicada de regiões em três dimensões (FLEISSNER; EBERHARD, 2008).

Figura 7 – Zona de interface entre processos.



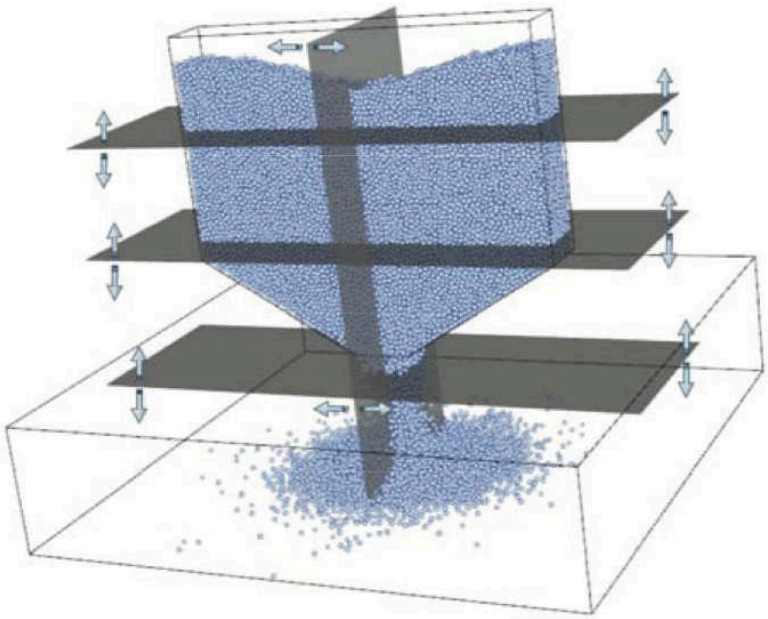
Fonte: Wang et al. (2004).

Koradi, Billeter e Güntert (2000) por exemplo utilizam, em dinâmica molecular, um método de particionamento e balanceamento de carga baseado em células de Voronoi. Os centros das células são atualizados de acordo com o movimento do centro de massa das partículas associadas aos subdomínios. Já no trabalho de Fattebert, Richards e Glosli (2012), os centros das células são deslocados através um método de minimização de custo.

As técnicas de particionamento e balanceamento apresentadas anteriormente utilizam estratégias geométricas de decomposição de domínio. Nesses trabalhos a paralelização computacional é feita seguindo o padrão de troca de mensagens entre processos. Observe que alguns destas estratégias de particionamento são aplicados em problemas de interação de longo alcance entre partículas, como no caso de dinâmica molecular. No entanto, os conceitos apresentados podem ser importados para estudo de problemas de interação de curto alcance, a exemplo do trabalho de Fleissner e Eberhard (2008). No trabalho de Markauskas, Kaceniauskas e Maknickas (2011), o emprego de uma técnica simples e rápida baseada no movimento de planos, similar à apresentada por Kohring (1995), já conduz a resultados satisfatórios de balanço de carga. A Figura 8 ilustra o método de particionamento aplicado no estudo de um funil de descarga.

Outras técnicas de particionamento, não necessariamente geométricas, vem sendo empregadas tanto para ambientes de memória compartilhada (OWEN; FENG, 2001) como distribuída (OWEN et al., 2004). O trabalho de Owen e Feng (2001), por exemplo, utiliza técnicas de particionamento baseadas em topologia na análise paralela de sistemas discretos. A definição de topologia consiste na criação de grafos que consideram as partículas do modelo e as interações entre estes elementos. O processo de particionamento utiliza heurísticas para dividir o grafo tentando minimizar a comunicação depois da decomposição. Dois modelos são adotados com intuito de definição do grafo: *a priori* e *a posteriori*. O modelo *a priori* considera a lista de contatos atual das partículas, enquanto que o modelo *a posteriori* considera ainda a lista potencial de contatos para a definição da

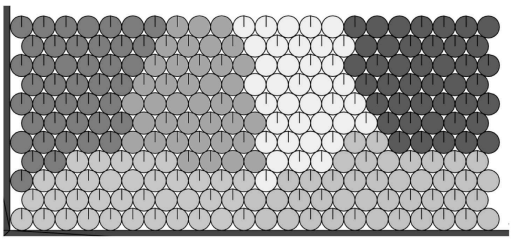
Figura 8 – Balanceamento de carga por movimento de planos.



Fonte: Markauskas, Kaceniauskas e Maknickas (2011).

topologia. O balanceamento dinâmico de cargas é ainda considerado no trabalho através da técnica de re-particionamento. Os métodos de particionamento topológico em geral produzem decomposições de melhor qualidade, mas em geral são mais custosos. A Figura 9 ilustra uma decomposição de domínio baseada em topologia apresentada no trabalho de Owen e Feng (2001). Os tons de cinza apresentados ilustram os subdomínios gerados pelo procedimento.

Figura 9 – Particionamento baseado em topologia.



Fonte: Owen e Feng (2001).

Embora o trabalho de Owen e Feng (2001) utilize a técnica de particionamento topológico de domínio como estratégia de paralelização computacional, a metodologia empregada no trabalho baseia-se em procedimentos de paralelização para ambientes de memória compartilhada. Diversas etapas envolvidas neste procedimento ocorrem de maneira mais direta e simplificada, a exemplo da montagem do grafo de conectividade, distribuição de partículas e tratamento das interfaces de comunicação entre processos. Posteriormente Owen et al. (2004) apresentam uma metodologia de paralelização que permite o uso desta

estratégia de particionamento em ambientes computacionais de memórias distribuídas.

A utilização das técnicas de particionamento topológicas para processamento distribuído exige uma metodologia mais complexa de comunicação e manipulação de estruturas de dados (WANG; FENG; OWEN, 2004). Um estudo comparativo de simulações paralelas do DEM foi realizado anteriormente considerando uma técnica topológica e uma geométrica (MARKAUSKAS; KAČENIAUSKAS, 2015). Para tanto, o método topológico de particionamento de grafo (k-way) (KARYPIS; KUMAR, 1999) e o método geométrico RCB foram confrontados no estudo de performance paralela de uma simulação de fluxo de partículas em um funil de descarga. As simulações computacionais realizadas demostram que a técnica geométrica mostra-se mais apropriada devido à sua facilidade de implementação e desempenho superior observados.

1.3 Objetivo

O objetivo deste trabalho é desenvolver e avaliar técnicas de particionamento e balanceamento de carga, em ambientes de memória distribuída e compartilhada, para a paralelização computacional do Método dos Elementos Discretos. As técnicas de particionamento adotadas baseiam-se em critérios geométricos, topológicos e em métodos combinados. São consideradas diversas condições de uso das técnicas de decomposição de domínio com o intuito de estimar a qualidade dos particionamentos obtidos.

Propõe-se ainda, como principal contribuição do trabalho, o desenvolvimento de uma metodologia de paralelização baseada na curva de preenchimento de espaço de Hilbert (HSFC - *Hilbert Space Filling Curve*) (HILBERT, 1891). A técnica de solução proposta visa otimização de localidade de dados e ao mesmo tempo permite um procedimento de particionamento local aplicado em modelos de paralelização híbridos. Esta estratégia de solução é dotada de mecanismos que permitem um balanço de carga tanto em nível de processamento distribuído como de definição de trabalhos computados por *threads*.

1.4 Contribuições

Este trabalho introduz uma estratégia híbrida de paralelização aplicada a simulações do DEM. A metodologia de solução alternativa proposta combina tarefas comumente necessárias na paralelização do DEM. A estratégia proposta reduz custos computacionais associados com os métodos de particionamento e simplifica tarefas associadas.

As implementações computacionais são realizadas no programa *DEMOOP* (*Discrete Element Method Object Oriented Programming*) (JÚNIOR; CINTRA; SILVEIRA, 2006), sistema que tem recebido contínuo aperfeiçoamento e com aplicações práticas de interesse

do setor de Petróleo e Gás Natural. Durante o desenvolvimento deste trabalho surgiram algumas contribuições mencionadas a seguir:

- Desenvolvimento de uma metodologia de paralelização com particionamento baseado em topologia aplicada a simulações através do Método dos Elementos Discretos em ambientes de computação paralela distribuída;
- Otimização computacional da estrutura de armazenamento de dados de contato empregadas no DEMOOP;
- Unificação dos sistemas computacionais DEMOOP e DEMPad (SENA; SILVEIRA, 2007), em uma única versão paralela distribuída para facilitar a manutenção do código e expansão de funcionalidades;
- Implementação computacional de técnicas de pré-processamento baseadas em estruturas cristalinas, separação geométrica e minimização de distâncias;
- Realização de testes de performance em larga escala (123 milhões de partículas em até 256 processos) para a versão unificada e com particionamento de domínio baseada em faixas;
- Implementação de mecanismos de pós-processamento paralelo compatíveis com os sistemas computacionais ParaView (AHRENS; GEVECI; LAW, 2005a) e DEMView (AMORIM; CINTRA; LIRA, 2006);
- Emprego de técnicas de particionamento de domínio e balanceamento de cargas no *software* DEMOOP, através das bibliotecas ParMETIS (KARYPIS; SCHLOEGEL; KUMAR, 2003), METIS (KARYPIS; KUMAR, 1998a) e Zoltan (BOMAN et al., 2012);
- Desenvolvimento de uma metodologia que combina tarefas de controle de acesso à memória, particionamento local e balanceamento dinâmico de carga através de um modelo de paralelização híbrida baseada em HSFC.

Parte dos resultados apresentados neste trabalho foram submetidos a periódicos científicos onde a metodologia abordada e os resultados obtidos são discutidos.

1.5 Organização da Tese

Este trabalho está organizado em 11 capítulos, espalhados nas 3 partes que abordam revisão bibliográfica, metodologia e resultados. No presente capítulo (Capítulo 1), encontra-se a introdução ao tema, seu correspondente estado da arte e os objetivos referentes ao desenvolvimento deste trabalho.

O Capítulo 2 apresenta o Método dos Elementos Discretos com alguns detalhes de sua formulação, técnicas de geração de partículas, integração temporal e busca de contato.

No Capítulo 3, algumas técnicas de particionamento de domínio baseados em geometria e grafos são apresentadas. Técnicas de particionamento multi-nível, algoritmos de particionamento paralelo e estratégias de balanceamento dinâmico de carga também são abordados.

No Capítulo 4 introduz-se o sistema computacional empregado como base para as implementações computacionais desenvolvidas. Apresentam-se ainda as otimizações de código serial realizadas e estratégias de controle de acesso à memória.

O Capítulo 5 apresenta as técnicas de paralelização distribuída utilizadas no trabalho, enquanto que o Capítulo 6 introduz as estratégias de paralelização híbridos.

O Capítulo 7 aborda os resultados da estratégia de paralelização baseada em faixas, o Capítulo 8 trata das estratégias de particionamento topológicos e o Capítulo 9 discute a estratégia de particionamento RCB.

Por fim, no Capítulo 10 apresentam-se os estudos relacionados com as estratégias de paralelização híbridas e, no Capítulo 11, as conclusões do trabalho.

2 Método dos Elementos Discretos

O procedimento básico do DEM consiste em simular computacionalmente a dinâmica de um dado conjunto de partículas. A dinâmica destas entidades é regida segundo princípios básicos da física, basicamente as equações de movimento de Newton. As forças atuantes nas partículas são decorrentes de ações externas do meio em que estas se encontram inseridas, bem como de ações interativas despertadas por contatos ou mesmo por coesão (POTYONDY; CUNDALL, 2004).

Em geral, as principais tarefas de processamento do algoritmo do método das partículas são a busca de contato entre elementos discretos, o teste para verificar intersecção geométrica entre partículas, o cálculo de forças de interação e a integração temporal da equação de movimento de corpos rígidos. Tratam-se de procedimentos de natural paralelização computacional, uma vez que são realizados sobre um conjunto de dados independentes.

Adicionalmente, o DEM oferece desafios relacionados com as etapas de pré e pós processamento de modelos de grande porte. A geração geométrica, por exemplo, de um conjunto inicial de partículas para simulação numérica é um processo que demanda tempo de processamento significativo. Algumas técnicas de geração utilizam malhas de elementos finitos ou mesmo um processo de minimização de distâncias, utilizando o algoritmo de Levenberg-Marquardt (LEVENBERG, 1944; MARQUARDT, 1963), para definir o estado inicial de simulações de elementos discretos. Estas técnicas também apresentam tarefas naturalmente paralelizáveis, sejam por envolver um domínio de partículas ou por exigirem a solução de sistemas de equações lineares de elevadas ordens e esparsidade.

As questões envolvendo o pós-processamento gráfico também são consideráveis. A visualização de sistemas envolvendo grande número de partículas requer o uso de recursos gráficos avançados, tais como programação em placa gráfica e renderização distribuída, dentre outros (VENETILLO; CELES, 2007).

2.1 Breve histórico

A Mecânica Computacional do Descontínuo é apresentada por Munjiza (2004) como uma área relativamente nova de pesquisa. Nesta área de conhecimento, lidam-se com soluções de problemas de engenharia onde leis constitutivas não estão disponíveis. Utilizam-se ao invés disso modelagens numéricas baseadas em partículas, onde o comportamento micro-estrutural dos elementos do material é empregado. Para tanto, milhões ou mesmo bilhões de partículas são utilizadas com o intuito de reproduzir propriedades físicas com

importância prática. Métodos numéricos da mecânica do descontínuo incluem o Método dos Elementos Discretos, Métodos de Análise de Deformação Descontínua (DDA - *Discontinua Deformation Analysis Methods*), Métodos de Dinâmica Molecular, dentre outros.

O trabalho que formalizou o DEM é de autoria de Cundall e Strack (1979). No trabalho de Cundall e Strack, um modelo numérico capaz de descrever o comportamento mecânico de um conjunto de discos ou esferas, na ocasião chamado de *Método dos Elementos Distintos*, foi apresentado.

Somente mais tarde, Cundall e Hart (1992) definem o *Método dos Elementos Discretos* como um método computacional capaz de calcular deslocamentos e rotações finitas completamente independentes para corpos discretos e reconhecer novos contatos automaticamente à medida em que a análise ocorre.

A denominação *Método dos Elementos Discretos*, engloba portanto uma família de métodos numéricos comumente empregada na análise de meios descontínuos. Apesar de o *Método dos Elementos Distintos* ser um caso específico da família apresentada por Cundall e Hart, é comum na comunidade científica que a nova denominação seja empregada nos dois casos.

2.2 Geração de partículas

Uma das tarefas essenciais para a simulações envolvendo o DEM é a de geração de partículas, que consiste na definição de posições geométricas e raios para o dado conjunto de partículas que representam um meio físico. Uma diversidade de estratégias tem sido abordada na geração de partículas com formas bem conhecidas, principalmente geometrias circulares e esféricas, para os casos bidimensionais e tridimensionais respectivamente. Essas estratégias podem ser subdivididas em dois grupos distintos: os algoritmos geométricos e os algoritmos dinâmicos.

2.2.1 Procedimentos geométricos

Os algoritmos geométricos são caracterizados pela utilização de técnicas puramente geométricas para gerar partículas em um determinado domínio. Esta característica em muitos casos reduz sensivelmente o tempo de processamento demandado na geração, uma vez que não demanda a solução de equações de movimento. O trabalho de Cui e O'Sullivan (2003) apresenta uma técnica geométrica de geração de partículas baseada em uma malha inicial de elementos triangulares ou tetraédricos. A técnica consiste em definir as posições das partículas de forma que as mesmas estejam inscritas no interior dos elementos da malha. Um fator limitante do emprego da técnica é a necessidade da malha inicial de elementos e a dificuldade do controle do tamanho das partículas.

Utilizando o conceito de geração de malhas por avanço de fronteira, Feng, Han e Owen (2003) apresentam uma técnica de geração onde as posições das partículas são calculadas com base em informações das partículas previamente incluídas no domínio. A técnica é bastante eficiente, mas apresenta algumas limitações em casos particulares de geração de modelos tridimensionais. Para garantir a robustez do algoritmo, neste caso particular, a complexidade do algoritmo cresce e acaba prejudicando a sua eficiência. Os mesmos autores apresentam posteriormente uma técnica de geração bastante simples denominada algoritmo de compressão (HAN; FENG; OWEN, 2005). Nesta técnica as partículas são geradas em posições aleatórias, sem que hajam sobreposições entre elas, e posteriormente movidas apropriadamente em uma direção de compressão.

Ainda na linha dos algoritmos geométricos, Labra e Onate (2009) apresentam uma técnica de geração de partículas em meios densos. A técnica de Labra e Onate é bem apropriada na geração de meios particulados densos, sendo descrita na subseção 2.2.1.2.

2.2.1.1 Geração por separação geométrica

A técnica de geração de partículas por separação geométrica é um procedimento de natureza iterativa aplicado em problemas onde se deseja prescrever o raio das partículas. Trata-se de um processo simples, com fácil implementação paralela, e que envolve somente parâmetros geométricos.

Este método gera inicialmente posições aleatórias para as partículas (dentro do domínio de geração) e, através de um procedimento iterativo, minimiza as sobreposições existentes entre as partículas através da aplicação de deslocamentos incrementais. A posição de uma partícula arbitrária i é definida como:

$$\boldsymbol{p}_i = \{x_i, y_i, z_i\} \tag{2.1}$$

onde x_i, y_i e z_i são as coordenadas cartesianas. Sua evolução ao longo do método iterativo segue portanto a forma:

$$\boldsymbol{p}_i^{n+1} = \boldsymbol{p}_i^n + \boldsymbol{\delta}_i \tag{2.2}$$

onde n indica o índice do passo iterativo e $\boldsymbol{\delta}_i$ é o correspondente deslocamento calculado.

O cálculo dos deslocamentos somente leva em consideração os pares de partículas em contato obtidos para o conjunto de partículas que contém um número n_p de elementos. De cada par de partícula em contato, $[i, j]$, e correspondente sobreposição geométrica, ϵ_{ij}, surgem incrementos de deslocamento. Estes incrementos são calculados de acordo com o vetor normal ao plano de contato e com os raios das partículas, r_i e r_j, na forma:

$$\boldsymbol{\delta}_i = \sum_{j=1, j\neq i}^{n_p} \phi_{ij} \frac{\epsilon_{ij}}{2 \, \|\boldsymbol{p}_i - \boldsymbol{p}_j\|} (\boldsymbol{p}_i - \boldsymbol{p}_j), \tag{2.3}$$

sendo

$$\epsilon_{ij} = r_i + r_j - \|\boldsymbol{p}_j - \boldsymbol{p}_i\|. \tag{2.4}$$

Os elementos da matriz ϕ_{ij} assumem valor unitário caso exista contato entre as partículas do par $[i, j]$. Em caso contrário, o valor nulo é assumido.

A Equação 2.3 pode ainda ser modificada de forma a considerar a existência de obstáculos. Estes elementos podem representar, por exemplo, os limites geométricos do domínio de geração. Assim sendo, a Equação 2.3 pode ser escrita como:

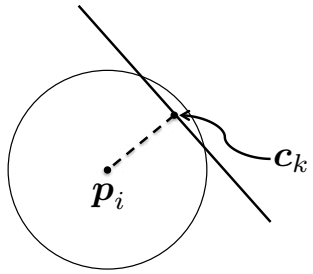
$$\boldsymbol{\delta}_i = \sum_{j=1, j\neq i}^{n_p} \phi_{ij} \frac{\epsilon_{ij}}{2\,\|\boldsymbol{p}_i - \boldsymbol{p}_j\|}(\boldsymbol{p}_i - \boldsymbol{p}_j) + \sum_{k=1}^{n_o} \phi_{ik} \frac{\epsilon_{ik}}{\|\boldsymbol{p}_i - \boldsymbol{c}_k\|}(\boldsymbol{p}_i - \boldsymbol{c}_k), \tag{2.5}$$

onde ϵ_{ik} representa o valor da sobreposição existente entre partícula e obstáculo para o par $[i, k]$. Seu valor é dado por:

$$\epsilon_{ik} = r_i - \|\boldsymbol{c}_k - \boldsymbol{p}_i\|, \tag{2.6}$$

onde $\boldsymbol{c}_k$ representa as coordenadas do ponto de contato. O ponto de contato $\boldsymbol{c}_k$ é ilustrado na Figura 10.

Figura 10 – Contato partícula-obstáculo e o ponto $\boldsymbol{c}_k$.



Os elementos da matriz ϕ_{ik} assumem o valor um caso existam contato para os pares $[i, k]$ (partícula-obstáculo). Em caso contrário, o valor zero é assumido. O número de obstáculos considerado na Equação 2.5 é representado por n_o.

O procedimento iterativo deve ocorrer até que a máxima sobreposição seja menor que um valor de tolerância estipulado, ou um número máximo de passos de solução seja atingido. A taxa de convergência do algoritmo é, em geral, maior nos casos onde a porosidade do meio é maior. O valor de tolerância para a sobreposição pode ser definido como um percentual do raio médio das partículas do conjunto gerado.

As posições das partículas não devem ser atualizadas no método de marcha da Equação 2.2, antes do cálculo dos deslocamentos de suas partículas vizinhas. Isso facilita a paralelização computacional uma vez que o cálculo de $\boldsymbol{\delta}_i$ pode ser feito de maneira completamente independente.

2.2.1.2 Geração por minimização de distâncias

A ideia básica do algoritmo é modificar um arranjo inicial de partículas, reposicionando ou redimensionando as partículas, através de um processo de minimização de

distâncias. Este último procedimento utiliza a estratégia de minimização de Levenberg-Marquardt (LEVENBERG, 1944; MARQUARDT, 1963).

O procedimento de minimização descrito a seguir objetiva a determinação de novas posições e raios para as partículas inicialmente dispersas no domínio. Este processo deve ocorrer de maneira a minimizar a distância entre as partículas do domínio e a impedir o surgimento de sobreposições geométricas. As variáveis envolvidas na estratégia de minimização são as coordenadas espaciais das n_p partículas envolvidas e seus respectivos raios. De uma forma global, o método de minimização determina o conjunto:

$$\boldsymbol{x} = \{\boldsymbol{p}_1^*, \boldsymbol{p}_2^*, ..., \boldsymbol{p}_i^*, ..., \boldsymbol{p}_n^*\}, \tag{2.7}$$

onde $\boldsymbol{p}_i$ contém as coordenadas espaciais das partículas (x_i, y_i, z_i) e seus respectivos raios (r_i), na forma:

$$\boldsymbol{p}_i^* = \{x_i, y_i, r_i\} \quad \text{(partículas bidimensionais)} \tag{2.8}$$

ou,

$$\boldsymbol{p}_i^* = \{x_i, y_i, z_i, r_i\} \quad \text{(partículas tridimensionais)} \tag{2.9}$$

dependendo da dimensão espacial empregada no modelo.

O cálculo das posições das partículas é feito de maneira a obter uma configuração de $\boldsymbol{x}$ que minimize a soma das distâncias entre as partículas localizadas em uma vizinhança espacial (incidentes). Desta forma, o procedimento minimiza o volume de vazios para o meio particulado obtido. Trata-se portanto de um problema de minimização sem restrições, onde se deseja resolver:

$$\min f(\boldsymbol{x}) = \sum_{i=1}^{n_p} q_i \tag{2.10}$$

para

$$q_i = \sum_{j=1, j\neq i}^{n_p} \Phi_{ij} E_{ij} \tag{2.11}$$

e,

$$E_{ij} = ||\boldsymbol{p}_i - \boldsymbol{p}_j|| - (r_i + r_j) \tag{2.12}$$

onde Φ_{ij} é uma matriz cujos elementos assumem valores unitários nos pares $[i, j]$ incidentes e nulos nos demais, sendo os raios das partículas do par $[i, j]$ representados por $[r_i, r_j]$.

A solução proposta para a Equação (2.10) é o método de minimização de Levenberg-Marquardt. Neste método, as variáveis de projeto contidas no conjunto $\boldsymbol{x}$ (que inclui posições e raios das partículas) são obtidas na forma incremental através do expressão:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \boldsymbol{h}^k \tag{2.13}$$

em que o incremento $\boldsymbol{h}^k$ é obtido da solução do sistema de equações lineares:

$$\left[\boldsymbol{J}^T\boldsymbol{J} + \mu\boldsymbol{I}\right]\boldsymbol{h}^k = -\boldsymbol{J}^T\boldsymbol{q}. \tag{2.14}$$

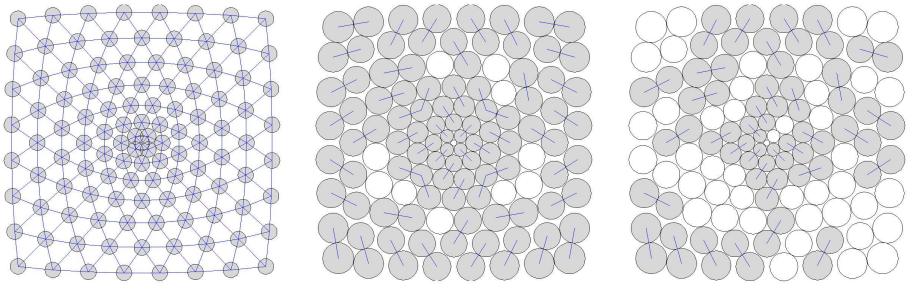
onde $\boldsymbol{I}$ é a matriz identidade e $\boldsymbol{J}$ é a matriz Jacobiana, dada por:

$$J_{ij} = \frac{\partial q_i}{\partial \boldsymbol{x}_j}. \tag{2.15}$$

O escalar μ é o parâmetro de amortecimento de algoritmo de Levenberg-Marquardt. Este parâmetro tem a função de assegurar que os incrementos obtidos estão em uma direção de descida, devendo assumir valores positivos.

O procedimento iterativo descrito na Equação (2.13) deve ser repetido até que os incrementos $\boldsymbol{h}^k$ apresentem valores abaixo de uma tolerância dada ou que o número de iterações k excedam um limite especificado. Recomenda-se que as matrizes E_{ij} e Φ_{ij} sejam armazenadas utilizando algum método que considere suas esparsidades, sobretudo para modelos de larga escala. A mesma consideração vale para a matriz resultante do sistema de equações lineares. Técnicas de resolução de sistemas de equações lineares podem ser utilizada para a solução paralela dos incrementos $\boldsymbol{h}^k$.

Figura 11 – Geração de partículas por minimização de distâncias: configuração inicial (esquerda), terceira iteração (centro), quinta iteração (direita).



As partículas realçadas em cinza representam partículas cuja incidência foi definida pelo critério de contato ou por uma malha inicial de referência. As partículas brancas são aquelas que não estão em contato ou com incidência definida.

2.2.1.3 Estruturas cristalinas

Uma prática bastante comum para geração de partículas é fazer uso de estruturas cristalinas. As estruturas cristalinas são redes que definem a distribuição espacial de átomos, íons ou moléculas em ligações químicas. Bravais (CAI; YANG; WANG, 2002) propôs que as estruturas cristalinas poderiam ser elaboradas com base em sete sistemas cristalinos básicos com base nos ângulo formado com 3 eixos de referência:

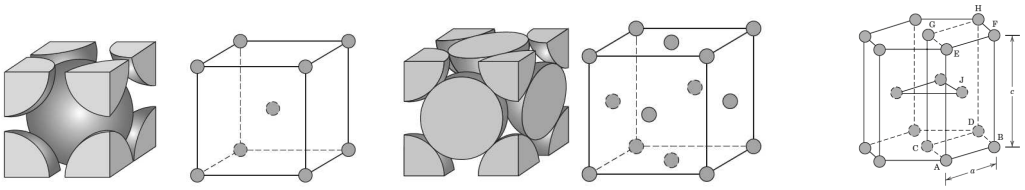
- Cúbico;
- Tetragonal;
- Ortorrômbico;
- Monoclínico;

- Triclínico;
- Hexagonal;
- Romboédrico.

Destes sistemas é possível compor 14 células unitárias, as quais englobariam qualquer tipo de estrutura cristalina conhecida. Dentre estas estruturas, algumas permitem um maior grau de empacotamento atômico, a exemplo da estrutura cúbica de corpo centrada (*Body Centered Cubic - BCC*), a cúbica de face centrada (*Face Centered Cubic - FCC*) e o *Hexagonal Closed-Packed - HCP*.

Os arranjos *BCC*, *FCC* e o *HCP* são comumente utilizados na geração de modelos de elementos discretos. Suas respectivas configurações geométricas estão ilustradas na Figura 12.

Figura 12 – Estruturas cristalinas.



(a) *BCC* (b) *FCC* (c) *HCP*

Fonte: Callister e Rethwisch (2013).

Uma das vantagens da geração de partículas através de sistemas cristalinos é que a determinação das coordenadas da rede é puramente geométrica e não requer procedimentos iterativos. É comum, no entanto, a utilização de partícula com raios idênticos.

2.2.2 Procedimentos dinâmicos

Os algoritmos dinâmicos são aqueles que utilizam a solução de uma equação de movimento como mecanismo de definição das posições das partículas. Esses algoritmos utilizam representações físicas semelhantes ou idênticas àquelas utilizadas na metodologia de solução do DEM (LIU; ZHANG; YU, 1999; SIIRIA; YLIRUUSI, 2007). Domínios particulados com geometrias arbitrárias podem ser gerados através de algoritmos dinâmicos, envolvendo ou não partículas de tamanhos variados.

Uma desvantagem desses algoritmos, no entanto, é o custo computacional. Em muitas situações, os parâmetros e modelos físicos empregados na geração exigem a solução de um grande número de iterações. Isso acontece, em muitos casos, em virtude de critérios relacionados à estabilidade de algoritmos de integração temporal. Em alguns casos, o fator limitante para a eficiência destes algoritmos é, além do número de partículas envolvidas na geração, a quantidade de passos de solução demandados.

2.3 Ciclo de cálculo

Segundo Cundall e Strack (1979), as partículas são entidades básicas para a representação de um meio discreto. Tratam-se de corpos com formas geométricas simples, sendo geralmente representados por discos ou esferas. A simulação com o DEM consiste em descrever a dinâmica de um meio descontínuo através das interações que ocorrem entre as partículas ao longo do tempo.

Para a análise numérica com o DEM, parte-se de uma configuração inicial do meio descontínuo, representado por partículas. Em um primeiro momento são definidas as posições e dimensões das partículas e em seguida, são aplicadas condições de contorno e condições iniciais, como restrições de deslocamentos e velocidades iniciais no modelo. Este procedimento é definido como uma etapa anterior à análise numérica com o DEM, conhecido como pré-processamento. Com a configuração inicial definida, parte-se para o início da simulação, que consiste num processo iterativo, onde o critério de parada por ser, por exemplo, quando se alcançar um tempo final de simulação ou quando ocorrer a estabilização energética do sistema.

Durante a simulação, as partículas são livres para se movimentar no espaço e têm dinâmica definida pelas leis de Newton, onde com o desenvolvimento de suas trajetórias, podem ocorrer eventuais choques entre partículas ou entre partículas e corpos contínuos. Portanto, é necessária a utilização de um mecanismo de busca de contatos para checar quais partículas estão interagindo em uma determinada iteração.

O contato entre as partículas é identificado a partir de um teste geométrico de intersecção, onde se avalia possíveis sobreposições entre corpos distintos. Este procedimento avalia, em determinada iteração, as posições de todo o conjunto de partículas que representam o meio em estudo.

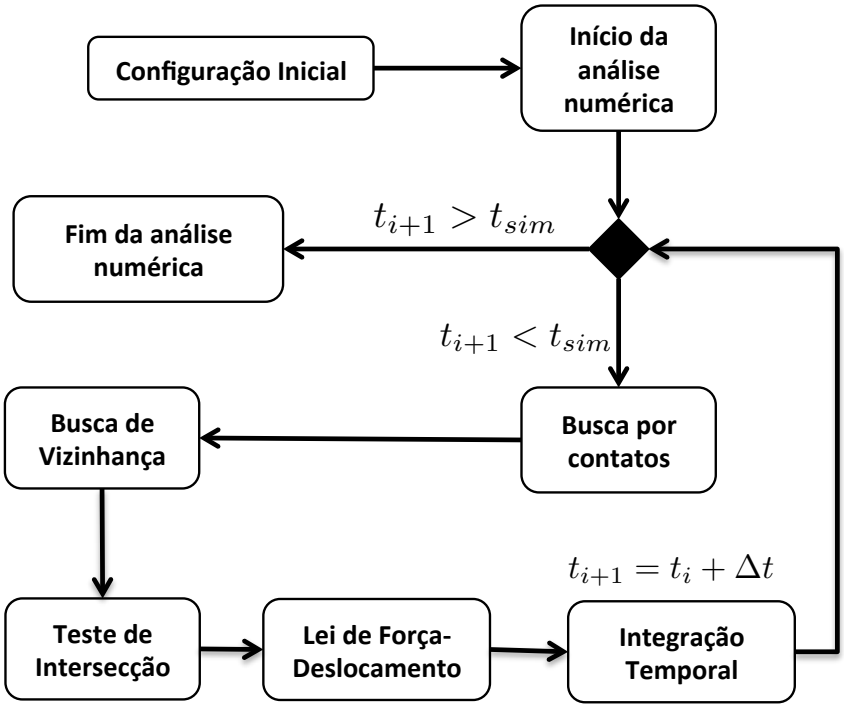
Assim como as partículas, outros corpos contínuos rígidos de geometrias arbitrárias também possuem dinâmica decorrente de suas forças atuantes. No caso da modelagem de problemas que envolvem a interação entre meios discretos e corpos contínuos, ou mesmo problemas de contato de outra natureza, essas forças atuantes envolvem componentes decorrentes de choques. Este fenômeno físico é representado através de modelos de contato, a partir da consideração de relações de força-deslocamento, seja para o choque entre partículas ou entre partículas e corpos contínuos rígidos. O choque entre partículas e corpos contínuos pode ser tratado de forma análoga àquela utilizada no tratamento de choques entre duas partículas, ou seja, este contato é detectado através de sobreposições de porções de corpos distintos e combatido através de forças resultantes que penalizam estas sobreposições.

Por fim, utiliza-se a dinâmica de corpos rígidos, onde, através do equilíbrio de forças do sistema, busca-se descrever os movimentos de translação e rotação das partículas

e corpos contínuos.

Com esse procedimento é possível determinar, e então atualizar, as posições dos corpos presentes no modelo. Verifica-se então se o critério de parada estabelecido foi atendido, levando ao fim da análise numérica. Caso contrário, procede-se com uma nova iteração, retornando-se a uma nova busca de contatos. A Figura 13 ilustra o procedimento comentado acima.

Figura 13 – Ciclo de cálculo do DEM.

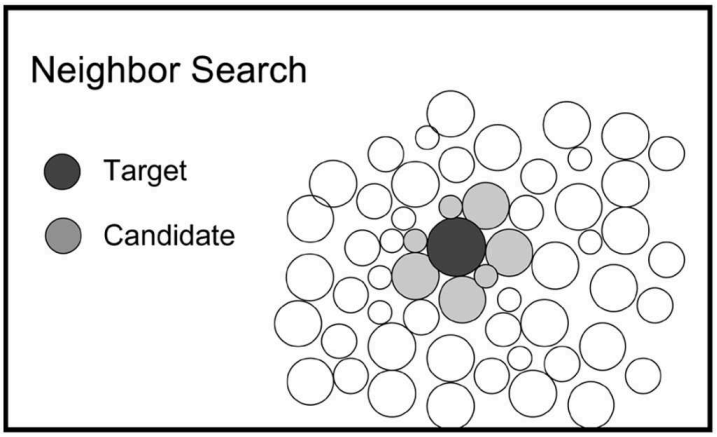


2.4 Busca de contatos

A busca por contatos consiste no processo de detecção de sobreposições geométricas para um dado conjunto de corpos dispersos no espaço correspondente ao domínio de busca. Este procedimento precede a etapa de aplicação das leis de força-deslocamento do DEM. Para realizar o procedimento de busca de contatos de forma eficaz, é necessária a utilização de um algoritmo que identifique apropriadamente as sobreposições geométricas dos corpos em questão. No DEM, em geral, os corpos possuem geometrias circulares ou esféricas (partículas) com o intuito de facilitar este processo de busca. A Figura 14 ilustra o cenário de busca de contatos para uma dada partícula do domínio (*Target*). Note que a partícula destacada é rodeada por diversas partículas, mas apenas algumas vizinhas próximas são passíveis de contato (*Candidate*).

Geralmente, a etapa de busca por contatos constitui uma das etapas críticas da simulação com o DEM. Para se ter ideia do alto custo computacional, supondo um sistema

Figura 14 – Busca de contatos.



Fonte: Perkins e Williams (2001).

com N partículas, em um algoritmo de verificação direta, cada uma dessas partículas irá testar a possibilidade de contato com cada uma das outras $N-1$ partículas, o que implica em um nível de complexidade computacional $O(N^2)$, se tornando inviável para uma quantidade razoavelmente grande de partículas. Com o objetivo de reduzir o custo computacional, buscam-se diferentes alternativas que procurem eliminar operações redundantes ou buscar contato entre corpos que estejam a um determinado limite de distância. Na literatura estão disponíveis vários algoritmos de busca por contatos. Entre estes se destacam alguns algoritmos que se baseiam na subdivisão espacial do domínio em células ou *grids*. Essas técnicas são cada vez mais utilizadas em vários tipos de aplicações devido a sua simplicidade e eficiência computacional.

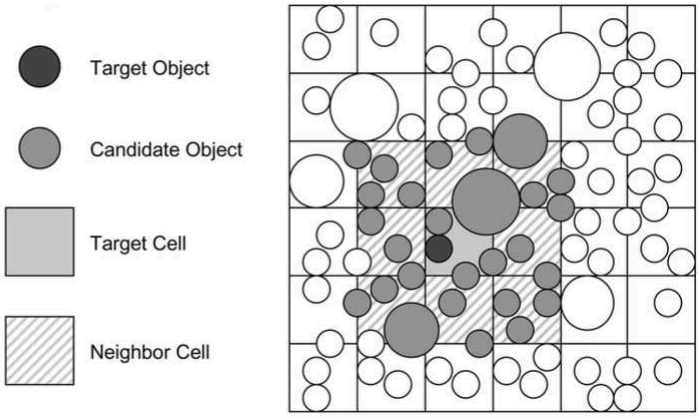
O algoritmo de busca de contatos denominado NBS (*No Binary Search*), proposto por Munjiza e Andrews (1998), se mostra bastante eficiente pois possibilita a busca de contatos num tempo de operação proporcional a N, onde N é o número de partículas do sistema. Algoritmos baseados em estruturas hierárquicas como árvores, também costumam ser eficientes. O algoritmo proposto por Feng e Owen (2002) tem complexidade computacional $O(NlnN)$. Um estudo comparativo de desempenho dos algoritmos de busca baseados em árvores e em células é apresentado por Han, Feng e Owen (2007). Neste trabalho opta-se pelo uso dos algoritmos baseados em *grids*, devido a sua boa eficiência.

Em geral os algoritmos de busca são divididos em duas etapas. Primeiramente, ocorre o mapeamento dos elementos discretos, associando-os as células em que estão contidos. Este procedimento tem como objetivo a remoção dos pares de corpos com sobreposição impossível ou improvável. Esta etapa é discutida em maiores detalhes na subseção 2.4.1. Em seguida, é feita a busca geométrica mais apurada por possíveis contatos, que podem ser entre partículas ou entre partícula e obstáculos (subseção 2.4.2).

2.4.1 Determinação da vizinhança das partículas

A ideia da técnica é criar uma estrutura de células que armazena informações referentes a partículas contidas em subdomínios espaciais. Através de uma função de mapeamento de coordenadas espaciais em subdomínios, é possível rapidamente inserir partículas ou consultar a existência destas em subconjuntos. A busca por partículas nestes subconjuntos, em geral, é bem mais eficiente computacionalmente do que no conjunto que contém todas as partículas do modelo (prática realizada no algoritmo de busca binária). Essa estratégia só é válida desde que seja respeitada a idealização de que a dimensão das células é dada pelo diâmetro máximo das partículas. A Figura 15 ilustra o procedimento de mapeamento das partículas na estrutura de células, destacando a partícula alvo da busca de contatos (*Target Object*) e suas vizinhas prováveis (*Candidate Object*), obtidas das células vizinhas. Além disso, as células de mapeamento tanto para a partícula alvo (*Target Cell*) como para as partículas vizinhas (*Neighbor Cell*) são apresentadas.

Figura 15 – Busca de contatos através de mapeamentos por *grids*.



Fonte: Perkins e Williams (2001).

A técnica de mapeamento em células é amplamente utilizada em programas de simulação por elementos discretos, e algoritmos de mapeamento e busca com propósitos diversos estão disponíveis na literatura (MUNJIZA; ANDREWS, 1998; WILLIAMS; PERKINS; COOK, 2004). O clássico algoritmo NBS utiliza a técnica de mapeamento de partículas em células. Este algoritmo tem uma boa eficiência e aplicação na busca de contatos entre partículas de tamanhos semelhantes. Quando existem partículas grandes inseridas em um universo maior de partículas predominantemente menores, o algoritmo apresenta a sua pior eficiência. Nesta situação existem muitas partículas mapeadas em uma mesma célula, o que torna o processo de busca mais lento.

Para a geração das células, define-se o volume envolvente do conjunto que contém as n_p partículas que seja deseja mapear espacialmente. Sendo x_i, y_i e z_i as coordenadas espaciais e r_i o raio da i-ésima partícula do domínio ($i = 1, 2, ..., n$), o volume envolvente

pode ser gerado a partir das coordenadas:

$$x_{\min} = \min\{x_i - r_i\}, \quad y_{\min} = \min\{y_i - r_i\}, \quad z_{\min} = \min\{z_i - r_i\}, \tag{2.16}$$

e

$$x_{\max} = \max\{x_i + r_i\}, \quad y_{\max} = \max\{y_i + r_i\}, \quad z_{\max} = \max\{z_i + r_i\}. \tag{2.17}$$

Na estratégia proposta no algoritmo NBS, as células que subdividem o domínio possuem arestas de dimensões definidas pelo máximo diâmetro das partículas do domínio. Logo,

$$d_{cell} = 2\max\{r_i\} \tag{2.18}$$

é o comprimento das arestas das células (quadrados ou cubos). Para o domínio que envolve as n_p partículas, geram-se, ao longo dos eixos coordenados, uma quantidade finita de divisões dadas por:

$$n_x = floor\left(\frac{x_{\max} - x_{\min}}{d_{cell}}\right) + 1, \tag{2.19}$$

$$n_y = floor\left(\frac{y_{\max} - y_{\min}}{d_{cell}}\right) + 1, \tag{2.20}$$

$$n_z = floor\left(\frac{z_{\max} - z_{\min}}{d_{cell}}\right) + 1. \tag{2.21}$$

onde $floor$ é a função de truncamento em números inteiros. Assim sendo, uma posição arbitrária (x, y, z) pode ser mapeada em um identificador de célula através da correspondência:

$$cell(x, y) = i_y(y)n_x + i_x(x) \quad \text{(casos bidimensionais)} \tag{2.22}$$

$$cell(x, y, z) = i_z(z)n_x n_y + i_y(y)n_x + i_x(x) \quad \text{(casos tridimensionais)} \tag{2.23}$$

onde,

$$i_x(x) = floor\left(\frac{x - x_{\min}}{d_{cell}}\right), \tag{2.24}$$

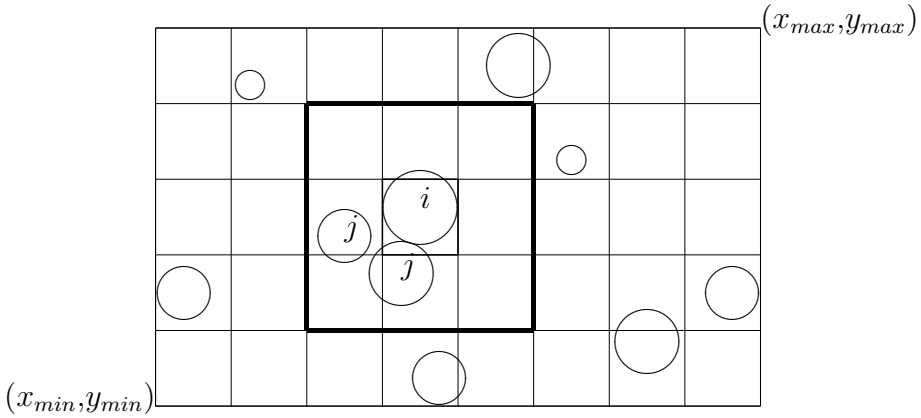
$$i_y(y) = floor\left(\frac{y - y_{\min}}{d_{cell}}\right), \tag{2.25}$$

$$i_z(z) = floor\left(\frac{z - z_{\min}}{d_{cell}}\right). \tag{2.26}$$

A função de mapeamento tem diversas aplicações, em geral, relacionadas à busca de vizinhança ou busca de contatos entre partículas. Neste caso, as partículas que interagem estão contidas na mesma célula ou em células imediatamente vizinhas (Figura 16). No caso específico ilustrado na Figura 16, a i-ésima partícula tem como vizinhas as j-ésimas partículas destacadas.

De forma geral, uma partícula localizada em uma posição arbitrária (x, y, z) tem como partículas vizinhas, e passíveis de contato, aquelas que pertencem às células que

Figura 16 – Mapeamento de partículas em células.



satisfazem:

$$\max\{0, i_x(x) - 1\} \leq i_x(x) < \min\{i_x(x) + 1, n_x\}, \tag{2.27}$$

$$\max\{0, i_y(y) - 1\} \leq i_y(y) < \min\{i_y(y) + 1, n_y\}, \tag{2.28}$$

$$\max\{0, i_z(z) - 1\} \leq i_z(z) < \min\{i_z(z) + 1, n_z\}. \tag{2.29}$$

Pode-se portanto, definir uma estrutura de dados que cria para cada célula espacial uma lista do conjunto de partículas contidas nesse subdomínio. Essa estrutura de dados acelera o processo de busca de vizinhança uma vez que torna implícita a informação de localidade das partículas. A complexidade da busca de vizinhança, para casos onde as dimensões das partículas são aproximadas, é linear. O custo da melhoria de eficiência de processamento, relativa à busca binária, no entanto é a exigência de memória adicional para armazenar a estrutura de células. Maiores detalhes sobre a implementação computacional da estrutura de células estão disponíveis no trabalho de Munjiza e Andrews (1998).

2.4.2 Teste de intersecção geométrica

A etapa de determinação de vizinhança define os pares de corpos com provável sobreposição geométrica. No entanto, esta etapa apenas precede o teste geométrico propriamente dito. A finalidade do teste de intersecção geométrica é definir se existem porções de dois elementos discretos ocupando o mesmo lugar no espaço. No DEM existem considerações que limitam estas sobreposições a pequenos valores, a fim de que princípios físicos não sejam violados. Por tratar-se de um método numérico baseado em penalidade, destas sobreposições calculadas surgem forças repulsivas para impedir o avanço da sobreposição (seção 2.6).

Deve-se ter em mente que o teste de intersecção é uma das etapas mais repetidas na estratégia de cálculo do DEM. O número de testes de intersecção realizado em uma simulação de elementos discretos é diretamente proporcional ao produto do número de partículas pelo número de passos de solução. Como estes dois últimos números são grandes,

em geral, fica evidenciada a necessidade de otimização das rotinas computacionais que implementam os cálculos envolvidos.

Diversas técnicas estão disponíveis na literatura para a realização do teste de intersecção geométrica. Estas técnicas consideram as mais diversas formas geométricas assumidas para os elementos discretos. Por simplificação de cálculo, os elementos discretos aqui tratados são partículas representadas por formas circulares ou esféricas. A condição necessária de intersecção para este caso, conforme ilustra a Figura 17, é satisfazer:

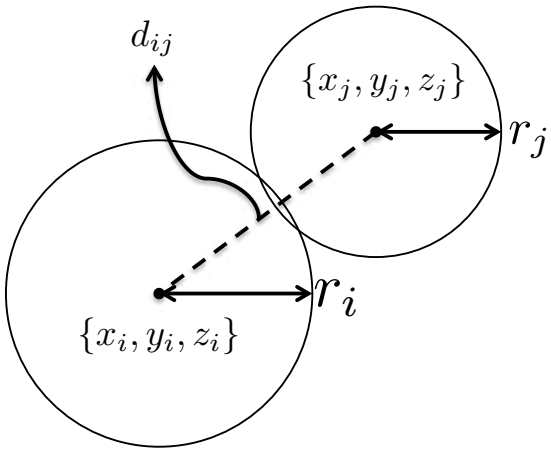
$$d_{ij} \leq r_i + r_j \tag{2.30}$$

onde d_{ij} é a distância entre as partículas i e j testadas, ou seja:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}. \tag{2.31}$$

sendo x, y e z as coordenadas nos eixos cartesianos. Os raios das partículas i e j são respectivamente r_i e r_j.

Figura 17 – Condição de contato entre partículas.



2.4.3 Busca de contato em domínios esparsos

Em algoritmos baseados em células, tais como o NBS, a estrutura de dados auxiliar responsável pelo mapeamento das partículas pode demandar uso excessivo de memória. Este problema acontece basicamente para problemas onde o domínio de solução envolve partículas bastante espalhadas. Em geral, os procedimentos de mapeamentos relacionados utilizam mecanismos que preenchem as células do domínio com informações auxiliares que armazenam referências para as partículas. A implementação destes mecanismos demanda alocação de memória até mesmo para as células em que não existe referência para uma partícula. Quando o número de células vazias é significativa, a quantidade de memória adicional requerida torna-se um problema. Às vezes, as células vazias nem mesmo são utilizadas durante o procedimento de busca dependendo da distribuição espacial das

partículas. Este problema acontece frequentemente em modelos de larga escala quando o domínio está altamente espalhado, ou seja quando o volume total das partículas é relativamente menor que o volume da caixa limite que as contém.

A busca de contatos para domínios espalhados deve ser realizada utilizando técnicas alternativas que não demandem o uso excessivo de memória. Uma alternativa consiste em utilizar procedimentos de busca binária para vetores ordenados por índices de células espaciais. Esta estratégia de indexação é também baseado em um *grid* regular de subespaços. Ao contrário do algoritmo NBS, a estratégia demanda uma estrutura de dados com tamanho do número de partículas. Para cada partícula, atribui-se um índice de célula correspondente nesta estrutura (vetor). O índice de célula da partícula é calculado a partir da posição deste elemento discreto e dos dados geométricos da caixa limite que contém todo o domínio. Para um dado conjunto de partículas, esta caixa limite é definida por dois pontos $b_k^{\min}$ e $b_k^{\max}$, dados por

$$b_k^{\min} = \min(x_{ik} - r_i), \tag{2.32}$$

e

$$b_k^{\max} = \max(x_{ik} + r_i), \tag{2.33}$$

onde $k = 1...dim$, sendo $dim = 2$ para modelos bidimensionais e $dim = 3$ para o caso tridimensional. A posição da partícula i na direção coordenada k é representada por x_{ik} nesta notação. O número total de células em cada direção coordenada é calculado por:

$$nc_k = \text{floor}(\frac{1}{d_{cell}}(b_k^{\max} - b_k^{\min})) + 1, \tag{2.34}$$

e o índice de célula para a partícula i é portanto definido como:

$$cell_i = ind_{i1} + \sum_{k=2}^{dim} ind_{ik} \prod_{w=1}^{w=k-1} nc_w, \tag{2.35}$$

considerando

$$ind_{ik} = \text{floor}(\frac{x_{ik} - b_k^{\min}}{d_{cell}}). \tag{2.36}$$

Uma tarefa de ordenação é então realizada para o vetor $cell_i$ e um vetor auxiliar contendo referências para as partículas. Este procedimento gera uma sequência de grupos de partículas agrupada por índice de célula. A técnica é similar à apresentada por Perkins e Williams (2001), mas o procedimento de ordenação é realizado para apenas um vetor. Os vizinhos de uma dada partícula são aqueles contidos nos grupos de partículas com índices de célula específicos. As células espaciais próximas à da partícula investigada, assim como no NBS, definem estes grupos.

Observe que esta estratégia assegura localidade na busca de contatos da mesma forma que o algoritmo NBS. No entanto, a estratégia requer que um procedimento de

busca seja realizado para o vetor que contém os índices de célula das partículas. Esta busca tem complexidade $log_2(N)$ uma vez que utiliza um vetor ordenado como referência. Desta forma, a complexidade do algoritmo de busca é definida basicamente pelo procedimento de ordenação empregado.

2.5 Equação de movimento

Embora a cinemática de um conjunto de partículas seja algo de difícil obtenção por meios analíticos, sobretudo quando muitos elementos estão envolvidos, a hipótese que rege o movimento no DEM é bastante simples. Neste método, a segunda lei de Newton é adotada para descrever o movimento de cada um dos elementos discretos que compõe o meio descontínuo. Assim sendo, considerando uma partícula arbitrária i, com massa m_i e momento de inércia I_i, as equações de movimento translacionais e rotacionais são dadas respectivamente por:

$$m_i \frac{d^2 \boldsymbol{x}_i}{dt^2} = \boldsymbol{F}_i, \tag{2.37}$$

e

$$\boldsymbol{I}_i \frac{d^2 \boldsymbol{\theta}_i}{dt^2} = \boldsymbol{T}_i. \tag{2.38}$$

Estas equações compõem um conjunto de equações diferenciais ordinárias de segunda ordem no tempo, resolvidas de forma independente. Métodos explícitos de integração temporal são frequentemente empregados na solução e usualmente demandam a solução de um número elevado de passos de tempo. Maiores detalhes sobre as técnicas de integração temporal são descritas na seção 2.7.

A interação entre as partículas e o meio externo é idealizada através das ações de forças e torques atuantes, representadas pelos vetores $\boldsymbol{F}_i$ and $\boldsymbol{T}_i$ respectivamente. Estes vetores podem ser determinados de diversas maneiras, dependendo das hipóteses consideradas no modelo numérico adotado na solução.

É ainda comum separar as componentes de força na direção normal ($\boldsymbol{F}_n^{(c)}$) e tangencial ($\boldsymbol{F}_t^{(c)}$) para permitir que modelos de força-deslocamento distintos sejam utilizados em cada direção. Assumindo então que cada partícula i pode interagir com outras partículas ou obstáculos através de um contato (c), então $\boldsymbol{F}_i$ pode ser escrito como:

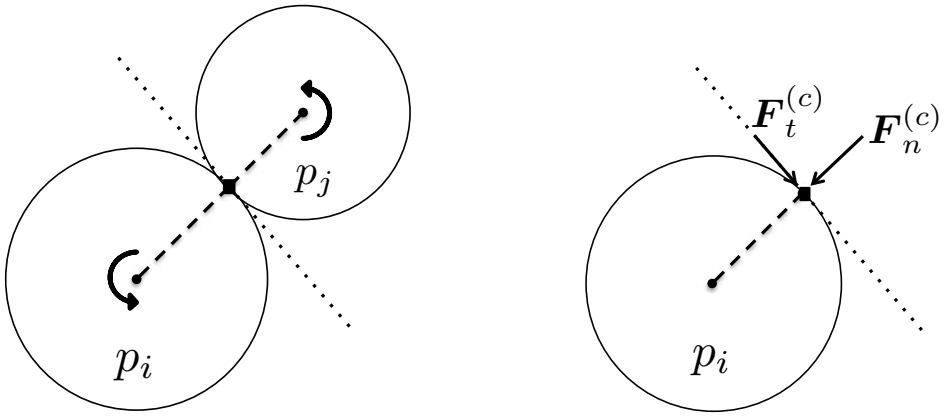
$$\boldsymbol{F}_i = m_i \boldsymbol{g} + \sum_{c=1}^{Nc} (\boldsymbol{F}_n^{(c)} + \boldsymbol{F}_t^{(c)}), \tag{2.39}$$

onde N_c é o número de contatos da partícula e $\boldsymbol{g}$ é a aceleração devido à gravidade. Os modelos de interação entre elementos discretos adotados neste trabalho são relatados na seção 2.6.

2.6 Modelos de interação

O fenômeno do contato é tratado no DEM através do surgimento de forças repulsivas que seguem leis de força-deslocamento diversas. Uma vez realizada a busca de vizinhança, conforme procedimento apresentada na subseção 2.4.1, e identificada uma sobreposição geométrica (subseção 2.4.2), a próxima etapa do ciclo de cálculo do DEM é o cálculo de forças interativas. Neste procedimento, define-se o plano de contato entre as partículas p_i e p_j, Figura 18 (a), e calculam-se as forças de penalidade em ambas as direções normal e tangencial ao plano. As forças obtidas são aplicadas no ponto de contato para as partículas p_i e p_j, conforme ilustra a Figura 18 (b), com mesma intensidade e direção mas com sentidos opostos.

Figura 18 – Interação entre partículas.



(a) Plano de contato (b) Forças interativas

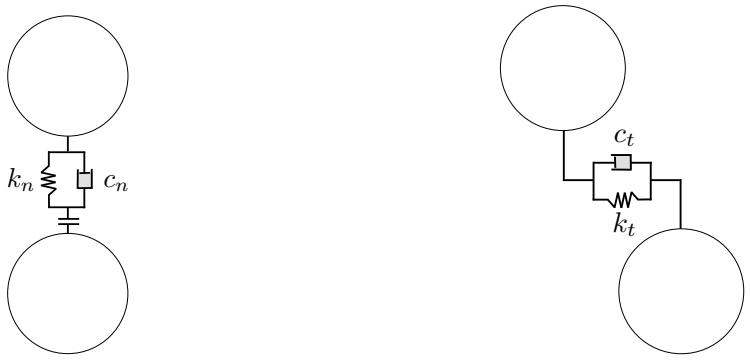
Diversas leis de força-deslocamento são apresentadas na literatura para o cálculo das forças de interação. Algumas destas leis consideram que apenas molas elásticas são responsáveis pela geração das forças repulsivas decorrentes dos contatos. Entretanto estes modelos não dispõem de mecanismos que permitam a dissipação de energia nos choques, logo os modelos viscosos são mais apropriados e utilizados (ONATE; ROJEK, 2004).

2.6.1 Modelo de Kelvin

O Modelo de Kelvin é composto por uma associação em paralelo de um elemento elástico linear com um elemento viscoso linear, conforme ilustra a Figura 19. A força calculada para este modelo é função não apenas dos deslocamentos relativos, mas também das velocidades relativas. O modelo considera a mesma formulação tanto para o cálculo da força normal quanto para o cálculo da força tangencial.

São definidos valores de rigidezes de contato normal k_n e tangencial k_t para as molas que visam impedir, respectivamente, a sobreposição das partículas e o deslocamento angular relativo do contato desde o seu início. Os parâmetros c_n e c_t correspondem a fatores de amortecimento, empregados com o intuito de dissipar a energia de movimento das

Figura 19 – Modelo de contato entre partículas.



(a) Direção normal (b) Direção tangencial

partículas. Estes parâmetros permitem a representação de choques não elásticos, bastante comuns em problemas físicos onde há perda de energia durante o choque. Logo, a força na direção normal obtida para duas partículas arbitrárias i e j em contato no evento (c), é dada por:

$$\boldsymbol{F_n^{(c)}} = k_n \ \boldsymbol{\delta}_n^{(c)} + c_n \ \boldsymbol{v}_{rn}^{(c)}, \tag{2.40}$$

enquanto que a força na direção tangencial assume a forma:

$$\boldsymbol{F_t^{(c)}} = k_t \ \boldsymbol{\delta}_t^{(c)} + c_t \ \boldsymbol{v}_{rt}^{(c)}, \tag{2.41}$$

onde $\boldsymbol{\delta}_n^{(c)}$ e $\boldsymbol{\delta}_t^{(c)}$ representam os deslocamentos relativos nas direções normal e tangencial do contato respectivamente. As velocidades relativas são representadas por $\boldsymbol{v}_{rn}^{(c)}$, na direção normal, e $\boldsymbol{v}_{rt}^{(c)}$ na direção tangencial, sendo obtidas a partir da diferença entre as velocidades das partículas do contato (c).

2.6.2 Modelo de Kelvin-Coulomb

Embora o modelo de Kelvin tenha sua aplicabilidade em diversos problemas, em geral, é comum o uso de um de seus modelos derivados para a representação de problemas que consideram o atrito. Modelos de força-deslocamento aplicados no contato entre elementos discretos comumente utilizam o modelo de Kelvin-Coulomb para a determinação das forças de contato na direção tangencial. Este modelo assume a existência do limite de atrito de Coulomb, além de uma mola e um amortecedor. O alongamento da mola define a componente elástica do contato, o amortecedor é responsável pela parcela viscosa e o limite de Coulomb tenta representar o fenômeno de atrito. Desta forma, a componente de força de contato na direção tangencial assume a forma:

$$\boldsymbol{F}_t^{(c)} = \begin{cases} \boldsymbol{F}_{trial}^{(c)} & \text{se } \|\boldsymbol{F}_{trial}^{(c)}\| \leq \mu \|\boldsymbol{F}_n^{(c)}\| \ , \\ \boldsymbol{F}_{fric}^{(c)} & \text{caso contrário,} \end{cases} \tag{2.42}$$

com

$$\boldsymbol{F}_{trial}^{(c)} = k_t \ \boldsymbol{\delta}_t^{(c)} + c_t \ \boldsymbol{v}_{rt}^{(c)}, \tag{2.43}$$

e

$$\boldsymbol{F}_{fric}^{(c)} = \mu \|\boldsymbol{F}_{n}^{(c)}\| \frac{\boldsymbol{F}_{trial}^{(c)}}{\|\boldsymbol{F}_{trial}^{(c)}\|}, \tag{2.44}$$

onde μ é o coeficiente de atrito de Coulomb.

2.6.3 Torque e atrito de rolamento

O torque atuante em uma dada partícula i pode surgir de ambas as forças normais e tangenciais definidas na interação de contato entre elementos. Isto acontece uma vez que as geometrias das partículas podem ser assumidas não perfeitamente circulares ou esféricas. Esta característica resulta em componentes de atrito que impedem o movimento de rolamento livre. Estas componentes de atrito são as responsáveis, por exemplo, pela redução da velocidade de partículas que rolam livremente em superfícies planas. Isto acontece uma vez que existe uma excentricidade entre o eixo de aplicação da força e o eixo de reação. Quando uma partícula é perfeitamente circular ou esférica, os dois eixos coincidem e não existe portanto este impedimento. A partícula, neste caso, não sofre acelerações decorrentes do atrito.

Considerando forças normais e tangenciais para um conjunto não perfeitamente circular ou esférico de partículas, pode-se definir o torque de contato na forma:

$$\boldsymbol{T}_i = -r_i \sum_{c=1}^{Nc} \boldsymbol{n}^{(c)} \times \boldsymbol{F}_{t}^{(c)} + \mu_r r_i \sum_{c=1}^{Nc} \|\boldsymbol{F}_{n}^{(c)}\| \frac{\boldsymbol{n}^{(c)} \times \boldsymbol{v}_{rn}^{(c)}}{\|\boldsymbol{n}^{(c)} \times \boldsymbol{v}_{rn}^{(c)}\|}, \tag{2.45}$$

onde $\boldsymbol{n}^{(c)}$ é o vetor que define a direção normal do contato e μ_r é o coeficiente de atrito de rolamento. Note que quando μ_r é nulo, uma das componentes da equação extingue-se decorrente da ausência do atrito de rolamento. Este modelo considera as velocidades relativas entre objetos no intuito de definir a direção do torque. Deve-se desconsiderar portanto, o cálculo da componente de atrito de rolamento quando $\|\boldsymbol{v}_{rn}^{(c)}\|$ for próximo de zero. Isto é necessário para que o procedimento de normalização utilizado não resulte em singularidades.

2.7 Métodos de integração direta

Os métodos de integração direta permitem a obtenção do histórico de resposta para a equação de movimento. Para tanto, a variável tempo é discretizada e a cinemática de um conjunto de elementos é obtida através um processo de integração passo-a-passo. Desta forma, os deslocamentos destes elementos são calculados em instantes de tempo separados por incrementos de tempo de integração (Δt).

A equação de movimento assume uma forma discreta comumente expressa por:

$$\boldsymbol{M}\ddot{\boldsymbol{u}}_n + \boldsymbol{f}_n^{int} = \boldsymbol{f}_n^{ext} \tag{2.46}$$

onde n representa o passo de tempo de integração e $\ddot{\boldsymbol{u}}_n$ corresponde ao vetor de acelerações. As forças internas e externas são representadas por $\boldsymbol{f}_n^{int}$ e $\boldsymbol{f}_n^{ext}$ respectivamente.

O processo de discretização no tempo é ainda acompanhado de aproximações por diferenças finitas para as derivadas temporais. O incremento de tempo de integração tem influência sobre a magnitude dos erros envolvidos com a utilização desta técnica aproximada. Usualmente, o valor do passo de tempo é fixo durante todo o processo de integração. Entretanto é possível utilizar estratégias de adaptação temporal com o intuito de controlar o tamanho do passo de tempo. Este procedimento de ajuste do incremento de tempo visa otimizar a relação entre erro e performance computacional (LAGES et al., 2013).

Os métodos de integração direta são classificados como explícitos ou implícitos dependendo da forma com que a resposta dos passos é obtida. Os algoritmos explícitos utilizam uma expressão na forma geral:

$$\boldsymbol{u}_{n+1} = f(\boldsymbol{u}_n, \dot{\boldsymbol{u}}_n, \ddot{\boldsymbol{u}}_n, \boldsymbol{u}_{n-1}, ...). \tag{2.47}$$

Observe que para a evolução da resposta para um passo qualquer $n+1$ são necessárias apenas informações de passos anteriores a este. Quando um algoritmo de integração direta faz uso de informações do passo $n+1$, esse é classificado como implícito. Logo, os algoritmos implícitos seguem a forma:

$$\boldsymbol{u}_{n+1} = f(\dot{\boldsymbol{u}}_{n+1}, \ddot{\boldsymbol{u}}_{n+1}, \boldsymbol{u}_n, \dot{\boldsymbol{u}}_n, \ddot{\boldsymbol{u}}_n, ...). \tag{2.48}$$

No processo incremental envolvido em um algoritmo de integração direta alguns aspectos devem ser observados. Em geral, estes aspectos estão associados a fatores tais como estabilidade e economia. Na prática, estes dois fatores constituem as principais diferenças entre os algoritmos explícitos e implícitos. A estabilidade pode ser entendida como sendo a capacidade de um algoritmo não amplificar erros provenientes de truncamentos numéricos ou de integrações imprecisas de modos de frequência alta. O termo economia, aqui apresentado, está relacionado com os custos das operações realizadas na avaliação de um passo de integração.

Os algoritmos de integração explícitos possuem estabilidade condicional. Isso significa que a estabilidade só é assegurada com o uso de um incremento de tempo de integração menor que um certo valor crítico. Quando essa exigência não é atendida o processo numérico se torna instável e a convergência não ocorre. Uma vez que o valor do incremento de tempo crítico Δt_{crit} é, em geral, bastante pequeno, muitos passos de integração são necessários no processo de integração numérica. No entanto, as operações efetuadas a cada passo de integração não são dispendiosas, permitindo que elas sejam obtidas de forma rápida.

Os métodos explícitos de integração ainda permitem que a resolução da equação de recorrência seja realizada de maneira desacoplada, ou seja, com avaliação independente para os seus termos. Para tanto requer-se que a matriz de massa esteja em sua forma diagonalizada. Esta consideração tem forte impacto positivo nas rotinas computacionais que implementam esta técnica. Mecanismos de paralelização computacional mais simples podem ainda ser utilizados de forma adicional no intuito de otimizar o processo de integração, sobretudo em modelos onde a discretização no espaço ocorre de maneira mais refinada (FAHMY; NAMINI, 1994).

Os métodos de integração implícitos, sob certas condições, possuem estabilidade incondicional. Isso significa que a estabilidade do algoritmo é garantida independentemente do valor do incremento de integração que esteja sendo utilizado. Hughes (1987) apresenta o conjunto de parâmetros para os quais os algoritmos da família de Newmark (subseção 2.7.1) são incondicionalmente estáveis. Isso, no entanto, não significa que o processo de integração implícita ocorra de maneira mais rápida do que o correspondente processo realizado por um método explícito, uma vez que os cálculos envolvidos em cada passo de integração são mais dispendiosos. Além disso, as matrizes envolvidas no processo de integração não são diagonais, o que acaba por consumir muito mais recursos de armazenamento.

O uso de métodos implícitos é também desfavorecido quando fortes não-linearidades estão presentes, tais como em problemas de propagação de ondas (contato e impacto) ou ainda em análises de estruturas elastoplásticas submetidas a carregamentos rápidos. O estudo destes casos exige a utilização de um incremento de tempo de integração pequeno. Desta forma, a eficiência do método de integração implícita acaba sendo comprometida. Para problemas inerciais, frequentemente chamados de problemas de dinâmica estrutural, o emprego de um método de integração implícito é mais apropriado. Nestes casos as baixas frequências dominam a resposta e os carregamentos atuantes variam mais lentamente, a exemplo dos carregamentos resultantes de um terremoto. Lowrie (2004) apresenta um estudo comparativo entre alguns algoritmos de integração implícitos, tais como Runge–Kutta e Crank–Nicolson. O trabalho apresenta ainda considerações acerca do incremento de tempo de integração frente a problemas não-lineares.

Belytschko e Hughes (1983) apresentam um estudo comparativo onde os métodos de integração explícitos e implícitos são confrontados frente ao número de multiplicações envolvidas em um passo de integração. São avaliados três exemplos correspondentes a casos de malhas unidimensionais, planas e tridimensionais. O estudo mostra que a relação de custo computacional, implícito por explícito, tende a crescer com a dimensão da malha. A razão de tal crescimento é em parte associada ao crescimento de forma quadrática dos termos das matrizes cheias utilizadas nos algoritmos implícitos.

2.7.1 Método das Diferenças Centrais

Em sua formulação clássica, o DEM utiliza o Método das Diferenças Centrais como algoritmo de integração temporal. Esta escolha é decorrente das altas frequências envolvidas no sistema devido à natureza de contato e impacto envolvida no problema. Trata-se de um algoritmo de fácil implementação e que possui variantes apresentadas por diversos autores.

Nos estudos de dinâmica, é comum que a Equação (2.46) assuma uma forma particular considerando o amortecimento e uma relação tensão-deformação que ocorra de maneira elástica e linear. Sua forma discretizada, e portanto incremental, é dada por:

$$\boldsymbol{M}\ddot{\boldsymbol{u}}_n + \boldsymbol{C}\dot{\boldsymbol{u}}_n + \boldsymbol{K}\boldsymbol{u}_n = \boldsymbol{f}_n^{ext}. \tag{2.49}$$

O Método das Diferenças Centrais pode ser encarado como uma particularização do algoritmo implícito de Newmark, cujas equações de recorrência são dadas por:

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + \Delta t\dot{\boldsymbol{u}}_n + \Delta t^2\left[\left(\frac{1}{2} - \beta\right)\ddot{\boldsymbol{u}}_n + \beta\ddot{\boldsymbol{u}}_{n+1}\right], \tag{2.50}$$

e,

$$\dot{\boldsymbol{u}}_{n+1} = \dot{\boldsymbol{u}}_n + \Delta t[(1-\gamma)\ddot{\boldsymbol{u}}_n + \gamma\ddot{\boldsymbol{u}}_{n+1}]. \tag{2.51}$$

Observe que o deslocamento é aproximado por uma expansão em série de Taylor com uma precisão de segunda ordem e que, quando se supõe $\beta = 0$, o algoritmo de Newmark torna-se explícito, em conformidade com a forma geral apresentada na Equação (2.47). A definição de $\gamma = \frac{1}{2}$ é, portanto, o que justifica a denominação do algoritmo como sendo de diferença central. O algoritmo do método das diferenças centrais, variante do algoritmo de Newmark, é apresentado na Figura 20.

No trabalho de Krysl e Belytschko (1998) outra variante para o método das diferenças centrais é apresentada. Esta variante toma como base o algoritmo de integração apresentado por Park e Underwood (1980) e faz uma correção iterativa da velocidade presente na equação de movimento.

2.7.2 Estabilidade

Conforme apresentado anteriormente, a estabilidade dos algoritmos de integração explícitos é condicionada ao uso de um incremento de integração inferior a um valor crítico (Δt_{crit}). Para a definição deste incremento leva-se em consideração o valor da mais alta frequência natural verificada na malha (diga-se conjunto de partículas) que discretiza o modelo estudado (ω_{max}). Da análise de estabilidade linear (BELYTSCHKO; HUGHES,

Figura 20 – Algoritmo do Método das Diferenças Centrais - variante de Newmark (1959).

Inicialização
a. Definição das condições iniciais do problema ($n = 0$) e montagem da matriz de massa.

Processo Iterativo
1. Cálculo da velocidade no passo n
$\dot{\boldsymbol{u}}_n = \dot{\boldsymbol{u}}_{n-1} + \frac{\Delta t}{2}\left[\ddot{\boldsymbol{u}}_n + \ddot{\boldsymbol{u}}_{n-1}\right]$
2. Cálculo dos deslocamentos no passo $n+1$
$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + \Delta t\dot{\boldsymbol{u}}_n + \frac{\Delta t^2}{2}\ddot{\boldsymbol{u}}_n$
3. Predição de velocidade no passo $n+1$
$\dot{\boldsymbol{u}}^p_{n+1} = \dot{\boldsymbol{u}}_n + \Delta t\ddot{\boldsymbol{u}}_n$
ou
$\dot{\boldsymbol{u}}^p_{n+1} = \left[\boldsymbol{u}_{n+1} - \boldsymbol{u}_n\right]/\Delta t$
4. Cálculo do vetor de forças desequilibradas
$\boldsymbol{f}^{des}_{n+1} = \boldsymbol{f}^{ext}_{n+1} - \boldsymbol{f}^{int}_{n+1} - \boldsymbol{C}\dot{\boldsymbol{u}}^p_{n+1}$
5. Cálculo da aceleração no passo $n+1$
$\boldsymbol{M}\ddot{\boldsymbol{u}}_{n+1} = \boldsymbol{f}^{des}_{n+1}$
6. Atualização do passo e tempo corrente
$n \leftarrow n+1$
$t \leftarrow t + \Delta t$

1983), pode-se demonstrar que este incremento é dado por:

$$\Delta t_{crit} = \frac{2}{\omega_{\max}} \tag{2.52}$$

onde ω_{max} é o máximo auto-valor (λ_{max}) da matriz $\boldsymbol{M^{-1}K}$, na forma:

$$\omega_{\max} = \sqrt{\lambda_{\max}}. \tag{2.53}$$

sendo $\boldsymbol{M}$ e $\boldsymbol{K}$, respectivamente, as matrizes de massa e rigidez global do sistema. Assume-se que o incremento de integração obtido na análise de estabilidade linear também engloba os casos não-lineares (O'SULLIVAN; BRAY, 2004).

Logo, o conhecimento de $\omega_{\max}$ permite que o processo de integração ocorra de forma apropriada. Na prática, o conhecimento exato deste valor é evitado, uma vez que requer que o problema de autovalor envolvendo as matrizes globais do sistema seja resolvido. Recorre-se, portanto, ao conhecimento de um valor aproximado. Uma forma de aproximar o valor de $\omega_{\max}$ é supor o maior valor de frequência natural dentre todos os elementos sem suporte que compõem a malha (BELYTSCHKO; HUGHES, 1983). A seguinte relação, cuja prova é dada no trabalho de Belytschko e Hughes (1983), é utilizada:

$$\lambda_{\max} \leq \sqrt{\lambda^e_{\max}} \tag{2.54}$$

onde $\lambda^e_{\max}$ é o máximo auto-valor da matriz $\boldsymbol{M^{e-1}K^e}$, dentre todos os elementos que compõem a malha ($\boldsymbol{M^e}$ e $\boldsymbol{K^e}$ são, respectivamente, as matrizes de massa e rigidez do elemento "e").

No caso específico de elementos discretos, para um sistema com um grau de liberdade, o incremento de tempo é dado simplesmente por:

$$\Delta t_{crit} = \frac{T}{\pi} = 2\sqrt{\frac{m}{K_{ef}}} \tag{2.55}$$

onde T é o período de vibração livre do grau de liberdade, m é a massa da partícula e K_{ef} é a rigidez efetiva que rege o movimento da partícula (O'SULLIVAN; BRAY, 2004). Em geral, simulações de elementos discretos costumam utilizar fatores de segurança que incidem sobre o valor de incremento de integração crítico calculado.

Devido às elevadas frequências encontradas em simulações do DEM, é comum ainda que a massa do sistema seja multiplicada por um fator de amplificação (THORNTON, 2000). Isso permite que a marcha ao longo do tempo obtida no processo de integração temporal ocorra com incrementos de tempo maiores, e exigindo um número menor de passos de solução. No entanto, este procedimento não é recomendado quando as respostas de alta frequência são importantes, ou seja, quando os efeitos de inércia não são desprezíveis.

3 Particionamento e Balanço Dinâmico de Carga

A proposta das técnicas de particionamento ou decomposição de domínio é definir subdomínios de um dado conjunto de entidades que represente este domínio. No caso de modelos de elementos discretos, estas entidades podem ser idealizadas como as partículas do modelo.

Quando a técnica de decomposição de domínio é a ferramenta utilizada em técnicas de paralelização computacional, espera-se que a mesma forneça um bom balanceamento de carga, bem como reduza a comunicação entre processos. O balanceamento de carga é atingido, na maioria dos casos, quando a quantidade de entidades contidas nos subdomínios é aproximadamente igual. No entanto, a comunicação entre processos depende da forma com que as entidades estão relacionadas entre si. No caso de simulações de elementos discretos, as relações entre partículas ocorrem para o evento de contato, que está associado com as posições espaciais das partículas.

Uma das formas de avaliar qualitativamente um particionamento é com base no perímetro (bidimensional) ou na área (tridimensional) das regiões de fronteira entre subdomínios. Quanto menores forem as regiões de interface entre subdomínios melhor, uma vez que existe uma associação direta entre fronteiras de subdomínios e comunicação entre processos.

Para modelos com características dinâmicas, exige-se que o particionamento seja refeito ou adaptado para ajustar-se à nova configuração do modelo, a exemplo da mudança de posição das partículas ao longo do tempo. Desta forma, as técnicas de decomposição podem ser estáticas ou dinâmicas, a depender da sua necessidade de reconfiguração.

As técnicas de particionamento dinâmico são indicadas no estudo de problemas de natureza dinâmica, problemas de impacto, dentre outros.

Deve-se atentar, no entanto, ao fato que decomposições consecutivas completamente independentes entre si, em geral, são substancialmente distintas (OWEN; FENG, 2001). Isto conduz a uma elevação da quantidade de comunicação entre processos, o que é indesejável quando se busca eficiência em técnicas de paralelização.

A maioria das técnicas de particionamento dinâmico de domínio podem ser classificadas em duas categorias gerais: geométricas e topológicas (OWEN; FENG, 2001). As técnicas geométricas exploram as informações de posição dos elementos para fazer a determinação dos subdomínios. Já as técnicas topológicas utilizam as interações entre os elementos para o particionamento. Em geral, os métodos topológicos decompõem melhor

o domínio que os geométricos, no entanto são mais custosos em termos computacionais. Há ainda uma forma combinada de particionamento que considera ambas as informações geométricas e topológicas.

3.1 Particionamento geométrico

3.1.1 Divisão em células

Esta técnica, em geral, costuma dividir o domínio espacial em subdomínios com formas idênticas: as células unitárias. Diversas formas geométricas podem ser utilizadas para a decomposição de domínios, sejam estes bidimensionais ou tridimensionais.

Dentre as formas mais empregadas no caso bidimensional estão as faixas, cuja interface entre subdomínios vizinhos ocorre em apenas uma direção (SENA; SILVEIRA, 2007), e os blocos (quadrados) (KOHRING, 1995), com interface entre subdomínios ocorrendo em duas direções (Figura 2). No caso bidimensional, as interfaces entre subdomínios são predominantemente definidas por segmentos de retas.

As células empregadas nos particionamentos tridimensionais são frequentemente faixas (paralelepípedos) ou cubos. Células unitárias baseadas em estruturas cristalinas, a exemplo do *BCC* e *FCC* (Figura 4), também são empregadas (STIJNMAN; BISSELING; BARKEMA, 2003). No caso tridimensional, a interface entre subdomínios é, em geral, definida por planos.

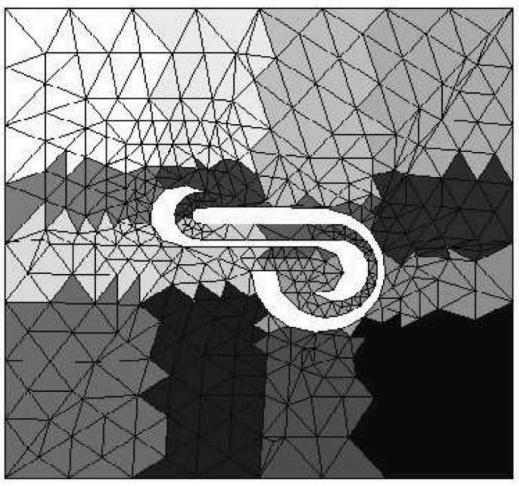
As técnicas de particionamento por divisão de domínio em células são de simples implementação. No entanto, estas técnicas desprezam dados de densidade de vértices e para muitas situações não geram subdomínios balanceados.

3.1.2 Bisseção de coordenadas (RCB e URB)

A técnica de bisseção recursiva de coordenadas (*Recursive Coordinate Bisection* - RCB) utiliza a informação de posição do conjunto de vértices que compõem um domínio para a sua divisão. O algoritmo, inicialmente proposto como estratégia de balanceamento estático por Berger e Bokhari (1987), divide o domínio na metade de acordo com as direções coordenadas. O processo é realizado levando em consideração o conjunto de vértices e selecionando as direções mais alongadas como critério de divisão. As metades obtidas são então divididas utilizando recursivamente o mesmo procedimento. O algoritmo RCB não é sensível a variações de densidade de vértices, uma vez que não considera as propriedades geométricas das sub-regiões resultantes. Esta técnica é bem empregada em situações onde os vértices estão uniformemente espalhados ao longo do domínio. Quando isto não ocorre, a técnica URB (*Unbalanced Recursive Bisection*) pode ser empregada para atingir melhores níveis de balanceamento (JONES; PLASSMANN, 1994). No algoritmo

URB, as formas geométricas resultantes são levadas em consideração para a secção dos planos de corte. A Figura 21 apresenta um exemplo de decomposição onde o algoritmo RCB foi empregado para a divisão de uma malha em 16 subdomínios (apresentados em tons de cinza distintos).

Figura 21 – Particionamento de domínio por RCB.



Fonte: Hu e Blake (2001).

Os algoritmos RCB e URB são bastante atrativos também para estratégias de balanceamento dinâmico de carga, uma vez que são incrementais. Isso significa que pequenas movimentações dos vértices resultam em pequenas movimentações dos planos de corte das sub-regiões (HENDRICKSON; DEVINE, 2000).

3.1.3 Bisseção inercial (RIB)

Um procedimento alternativo de particionamento de domínio similar ao RCB é o de bisseção inercial recursiva (*Recursive Inertial Bisection* - RIB), descrita por Simon (1991). Na técnica de bisseção inercial, os planos de corte não são previamente definidos como ortogonais aos eixos cartesianos, conforme ocorre no RCB. Ao invés disso, estes planos são definidos de acordo com o princípio mecânico de inércia. Para tanto, atribuem-se massas fictícias aos vértices e identifica-se a direção principal de inércia. O plano de corte é então definido como sendo ortogonal a esta direção. O procedimento de definição de plano e divisão de domínio é repetido recursivamente para as sub-regiões geradas. Embora este procedimento seja computacionalmente mais caro que as técnicas baseadas em coordenadas, os particionamentos resultantes são de melhor qualidade. O particionamento baseado em RIB, no entanto, não é incremental, uma vez que os eixos de inércia são mais sensíveis a perturbações.

3.1.4 Algoritmos baseados em índices

Muitos dos algoritmos de particionamento funcionam de maneira a gerar listas de vértices e então dividir estas listas em segmentos iguais. No entanto, existem técnicas clássicas para mapear um *grid* regular de dimensão d em uma lista unidimensional. Isso é feito de tal forma que vértices espacialmente próximos do *grid* d-dimensional sejam mapeados também próximos na lista unidimensional. Um *grid* irregular pode ainda ser mapeado em um *grid* regular e posteriormente mapeado na lista unidimensional. Dentre as técnicas baseadas em índices está a curva de Hilbert (HSFC). A vantagem dos algoritmos baseados em índices é que a ordenação pode ser calculada rapidamente, e que o algoritmo pode ser paralelizado. A qualidade dos particionamentos obtidos é comparável ao do algoritmo de bisseção de coordenadas. A biblioteca ParMETIS, por exemplo, faz uso de algoritmos baseados em índices na função ParMETIS_V3_PartGeom (KARYPIS; SCHLOEGEL; KUMAR, 2003).

3.2 Particionamento de grafos

Uma das razões que os algoritmos baseados em geometria tenham, relativamente, baixa qualidade é que eles não utilizam informações de conectividade entre elementos. Em virtude disto, uma série de algoritmos são desenvolvidos com o intuito de explorar esta informação. Logo, existe uma associação direta entre conectividade e grafos. Muito dos conceitos utilizados em técnicas de particionamento de domínio e balanceamento de carga podem ser, portanto, descritos usando a notação de teoria de grafos, resumida na subseção 3.2.1.

O problema de particionamento de grafos é decompor os vértices do grafo em p partições, aproximadamente iguais em número de vértices, tal que o número de arestas conectando diferentes partições seja minimizado (KARYPIS; KUMAR, 1998b). Este problema encontra diversas aplicações em muitas áreas incluindo computação científica paralela, agendamento de tarefas, projeto de circuito integrados (VLSI).

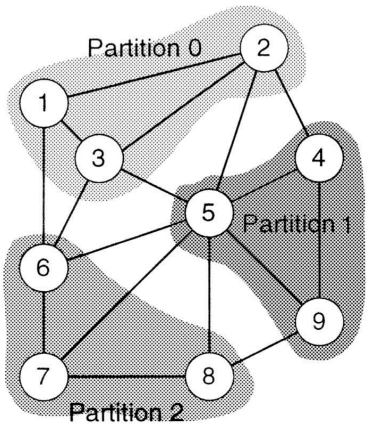
A Figura 22 ilustra o particionamento de um grafo com 9 vértices e 16 arestas em 3 partições ou subdomínios. No particionamento obtido os subdomínios foram subdivididos de tal forma que cada um deles contém 3 vértices e foram produzidos 9 cortes de arestas:

$$E_c = \{(1,6),(3,6),(5,6),(5,7),(5,8),(5,2),(5,3),(2,4),(8,9)\}. \tag{3.1}$$

3.2.1 Notação de teoria de grafos

Um grafo G pode ser entendido como um conjunto de componentes dentre os quais estão:

Figura 22 – Particionamento de grafos.



Fonte: Karypis e Kumar (1998b).

- Um conjunto de vértices V, entendido como uma lista de índices;
- Um conjunto de arestas E, onde cada aresta é composta por dois vértices.

O número de vértices de um grafo é representado por $|V|$, enquanto que o número de aresta é denotado por $|E|$.

Um grafo pode ser definido como direcionado, caso para uma determinada aresta exista início e fim, ou não direcionado caso não seja necessária a definição de sentido.

O grau do vértice pode ser entendido como o número de arestas do grafo conectada ao mesmo. Enquanto que o grau do grafo é o maior número dentre os graus dos vértices do conjunto V.

Um grafo é dito conectado se para quaisquer dois vértices i e j, denotados como $(i, j) \in E$ ou $i \leftrightarrow j$, existe um caminho que os conecta. Um caminho pode ser entendido como uma lista de vértices $\{i, i_1, i_2, ..., i_k, j\}$ tal que dois vértices consecutivos da lista formam uma aresta.

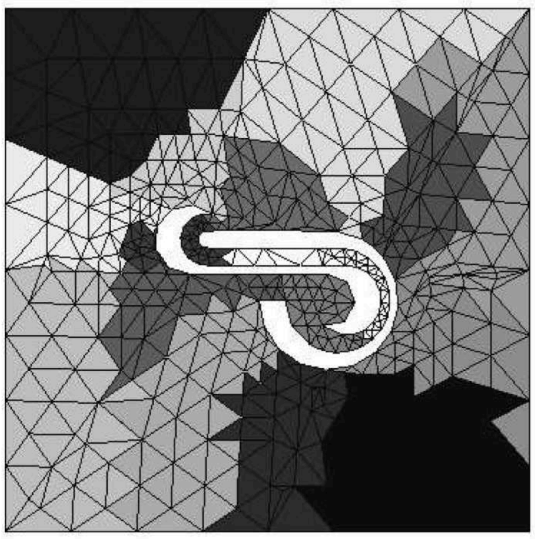
A distância entre dois vértices de um grafo é definida como o número mínimo de arestas dentre todos os caminhos que conectam os dois vértices. O conceito de distâncias entre vértices do grafo não deve ser confundido com a distância física entre dois pontos no espaço Euclidiano. Esta confusão de nomenclatura ocorre, muitas vezes, por existir uma associação entre vértices de grafo e coordenadas nodais de malhas ou posições de partículas.

3.2.2 Bisseção de grafo (RGB)

O algoritmo de bisseção recursiva de grafos (*Recursive Graph Bisection* - RGB) considera que o grafo a ser particionado esteja conectado. Determina-se inicialmente os dois vértices mais distantes entre si no grafo, esta distância é chamada de diâmetro do

grafo. Em seguida, iniciando de um deles (a raiz) seleciona-se a metade dos vértices que estão mais próximos para formar um subdomínio. Os vértices restantes formam o outro subdomínio. Este processo repete-se recursivamente para formar cada um dos subdomínios. O algoritmo de bisseção de grafo tem complexidade linear. Na Figura 23, apresenta-se uma decomposição gerada pelo algoritmo RGB para 16 subdomínios.

Figura 23 – Particionamento de domínio por RGB.



Fonte: Hu e Blake (2001).

3.2.3 Algoritmo Greedy

O algoritmo *Greedy* (guloso) é baseado na seleção de vértices com menores graus no grafo a ser particionado. Para um dado vértice selecionado, e sendo p o número desejado de subdomínios, o procedimento marca os primeiros $|V|/p$ vértices conectados. Os vértices marcados correspondem a um subdomínio e portanto são removido da lista de vértices a particionar. O mesmo procedimento de seleção, busca por conectividade e agrupamento repete-se para composição dos subdomínios restantes até que todos os vértices sejam marcados. Assim como o RGB, o algoritmo possui complexidade $O(|V|)$.

3.2.4 Bisseção espectral (RSB)

A técnica de particionamento por bisseção espectral recursiva (*Recursive Spectral Bisection* - RSB) formaliza um problema de otimização inteira para a definição dos cortes de arestas e definição dos subdomínios (POTHEN; SIMON; LIOU, 1990). Para cada vértice i do grafo são atribuídos valores x_i, seja 1 ou -1, representando as variáveis de projeto a serem determinadas. Define-se uma função objetivo que quantifica o número de cortes de arestas $|E_c|$, dada por:

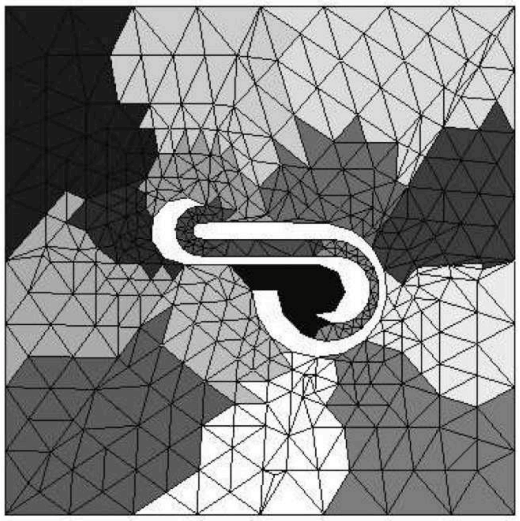
$$|E_c| = \frac{1}{4} \sum_{i \leftrightarrow j,\ i,j \in V} (x_i - x_j)^2 \tag{3.2}$$

sendo i e j dois vértices arbitrários conectados por uma aresta do grafo ($i \leftrightarrow j$). O procedimento de otimização deve minimizar o valor de E_c, alterando os valores de x_i. Quando $x_i \neq x_j$, para uma dada aresta $i \leftrightarrow j$, tem-se um corte de aresta. Os domínios particionados são formados pelos conjuntos dos vértices com mesmo valor para x_i. Para um bom balanceamento de carga é necessário satisfazer a restrição:

$$\sum_{i=1}^{n} x_i = 0. \tag{3.3}$$

Uma vez obtidos os subdomínios pela estratégia apresentada, repete-se o procedimento de maneira recursiva até o final da tarefa de particionamento. Em geral, o algoritmo RSB gera subdomínios conectados para um grafo conectado. Esta técnica de particionamento pode ter difícil solução, sobretudo se o número de vértices $|V|$ for grande, e exigir o cálculo de auto-valores e auto-vetores. Entretanto, a qualidade do particionamento gerado é significativamente superior a dos algoritmos RCB e RGB. A Figura 24 apresenta a decomposição de domínio obtida pelo algoritmo RSB, com cerca de 20% a menos de cortes de aresta que o obtido nos algoritmos RCB e RGB.

Figura 24 – Particionamento de domínio por RSB.



Fonte: Hu e Blake (2001).

3.2.5 Algoritmo K-L

O algoritmo K-L é um procedimento iterativo de particionamento de grafos inicialmente sugerido por Kernighan e Lin (1970). A estratégia parte de uma bisseção inicial balanceada e baseia-se no cálculo de uma função de ganho. Esta função considera a redução do número de cortes de arestas que ocorre quando um vértice é realocado de uma partição para outra. Para cada iteração, move-se o vértice com maior ganho da partição com mais vértices para a partição em déficit. Após este procedimento, o vértice é fixado e os ganhos são atualizados. Esta estratégia repete-se até que todos os vértices estejam fixos.

O algoritmo K-L tem características de estratégias de refinamento local, permitindo reduzir o número de cortes de arestas para um dado particionamento prévio. A bisseção inicial pode ser gerada aleatoriamente para grandes grafos, no entanto, os resultados obtidos são bastante dependentes da configuração inicial. Isso acontece porque o algoritmo tem dificuldade para sair de mínimos locais.

3.3 Estado da arte de algoritmos de particionamento

3.3.1 Algoritmos híbridos

Os algoritmos híbridos, ou combinados, são aqueles que utilizam algoritmos em conjunto para atingir melhores resultados. Esta combinação é feita com o intuito de aproveitar os melhores recursos que os algoritmos tem isoladamente, compondo-os em um novo algoritmo. A biblioteca ParMETIS por exemplo, faz uso de algoritmos híbridos em sua rotina ParMETIS_V3_PartGeomKway (KARYPIS; SCHLOEGEL; KUMAR, 2003). Nesta combinação, o algoritmo geométrico de preenchimento de espaço rapidamente calcula um particionamento inicial com base nas posições dos vértices do grafo, e, em seguida, o algoritmo multi-nível (ParMETIS_V3_PartKway) obtém o particionamento final de melhor qualidade.

Outro exemplo particular de combinação de algoritmos é o emprego do algoritmos K-L, apropriado para busca local, com algoritmos de otimização global e com os métodos multi-nível (KERNIGHAN; LIN, 1970; BUI; MOON, 1996; KARYPIS; KUMAR, 1998b).

3.3.2 Multi-nível

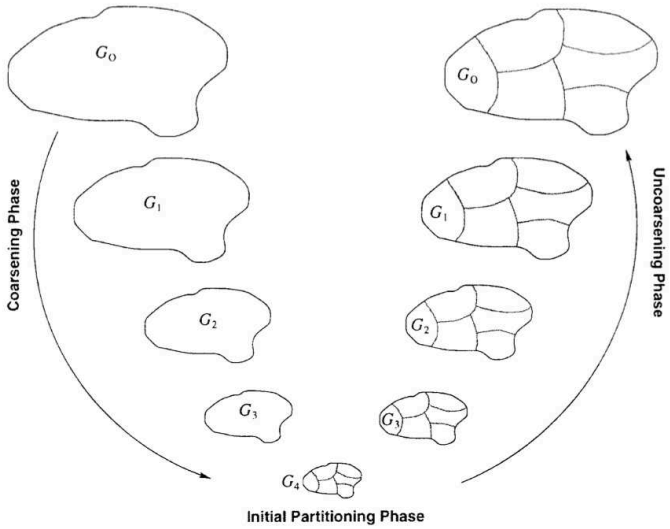
O objetivo dos algoritmos de particionamento multi-nível é encapsular a informação de conectividade de grafos G_0 através de sequências de grafos cada vez mais grosseiros $(G_1, G_2, ..., G_k)$. Em geral, os algoritmos baseados em multi-nível são divididos em 3 fases:

- redução: onde o grafo original G_0 é reduzido em uma série de grafos mais grosseiros $(G_1, G_2, ..., G_k)$;
- particionamento: onde o grafo mais grosseiro G_k é divido em p partes;
- interpolação e refinamento: onde o particionamento de G_k é interpolado para G_{k-1}, refinado e repetido em níveis até o grafo original G_0.

A Figura 25 ilustra as fases de redução (*coarsening*), particionamento (*partitioning*) e interpolação (*uncoarsening*).

O método mais comum para gerar grafos reduzidos é baseado em colapso de aresta, que atribui pesos para os vértices do grafo a fim de gerar sequências e particionamentos

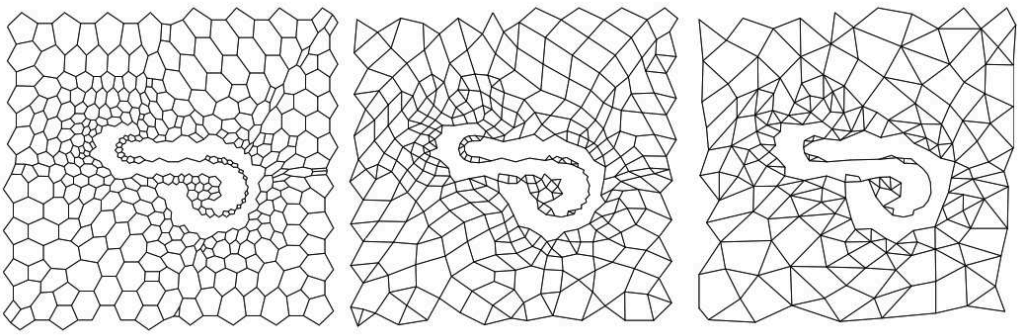
Figura 25 – Particionamento multi-nível.



Fonte: Karypis e Kumar (1998b).

de grafos reduzidos (HENDRICKSON; LELAND, 1995). A Figura 26 ilustra a estratégia de redução por colapso de aresta.

Figura 26 – Redução de grafo por colapso de arestas: G_0 (esquerda), G_1 (centro), G_2 (direita).



Fonte: Hu e Blake (2001).

Observe na Figura 26 que o número de vértices diminui de 788 vértices no grafo G_0 para 403 vértices no grafo G_1 e em sequência, no grafo G_2, para 210 vértices (26% no número de vértices do grafo G_0).

A biblioteca ParMETIS utiliza técnicas de particionamento de grafos em multi-nível através da função ParMETIS_V3_PartKway. O algoritmo de particionamento empregado, *k-way*, é descrito na sua forma sequencial no trabalho de Karypis e Kumar (1998b).

3.3.3 Algoritmos de particionamento paralelo

Embora as técnicas multi-nível reduzam significativamente o tempo de particionamento, estas técnicas exigem um uso intensivo de memória para decomposições em larga escala. Esta exigência excede, para simulações de larga escala, o limite de memória de uma única CPU (HU; BLAKE, 2001). Além disso, o propósito do procedimento de particionamento de domínio é geralmente o seu uso subsequente em sistemas computacionais paralelos. Em muitos casos, os dados necessários para a definição do particionamento estão espalhados no conjunto de processos responsáveis pela execução de uma dada tarefa, seja ela uma simulação numérica ou outro procedimento computacional.

Dentre as técnicas de particionamento paralela, destacam-se o algoritmo paralelo de bisseção recursiva multi-nível (PMRSB) e o algoritmo paralelo de particionamento multi-nível *p-way* (ParMETIS).

O PMRSB baseia-se no algoritmo de bissecção espectral multi-nível, sendo capaz de particionar um grafo com 250 mil vértices em 256 subdomínios até 140 vezes mais rápido que em seu processamento sequencial (BARNARD; SIMON, 1995).

Já o algoritmo paralelo de particionamento multi-nível *p-way*, empregado na biblioteca ParMETIS, modifica e paraleliza o algoritmo sequencial de particionamento *p-way* adotado no METIS (KARYPIS; KUMAR, 1998b; KARYPIS; KUMAR, 1999).

A implementação computacional de sistemas paralelos costuma demandar conhecimentos bastante específicos, o que tem motivado o uso de ferramentas computacionais dedicadas para uma rápida decomposição de domínio ou mesmo para um balanceamento dinâmico de carga. Diversos pacotes de particionamento estão disponíveis para acesso público, a exemplo do: CHACO, JOSTLE, METIS/ParMETIS, PARTY, PMRSB, S-HARP, SCOTCH, TOP/DOMDEC e WGPP (HU; BLAKE, 2001).

Em estudo comparativo de performance e qualidade de particionamento, o ParMETIS apresenta uma melhor performance, entre 25 e 30 vezes mais rápido, e produz particionamentos com uma melhor qualidade, aproximadamente 3 vezes menos cortes de aresta, que o PMRSB (KARYPIS; KUMAR, 1999). Além disso o PMRSB está disponível apenas para plataformas CRAY.

3.4 Balanceamento Dinâmico de Carga

As estratégias de particionamento de domínio em geral tentam equalizar a carga de trabalho dos processos responsáveis pela solução dos subdomínios. Esta condição é necessária para que haja um melhor aproveitamento dos recursos computacionais envolvidos. A obtenção de um bom balanço de carga computacional é uma tarefa complexa. Na teoria, esta tarefa deveria envolver o cálculo do número total de operações de ponto flutuante

durante um intervalo de tempo amostral. Uma forma simplificada de medir o balanço de carga no DEM é simplesmente comparando o número de partículas nos processos paralelos. Considerando esta aproximação razoável, o tempo de processamento de uma simulação do DEM é proporcional ao número de partículas utilizado.

Em geral, o número total de partículas em cada processo pode ser coletado e, de acordo com uma métrica, uma decisão determina se é necessária a redistribuição de partículas e definição de novos subdomínios. Uma forma simples de balanço de carga consiste em repetir o processo de particionamento. No entanto, esta prática pode gerar subdomínios substancialmente diferentes dos anteriores quando nenhuma informação histórica é considerada no particionamento (OWEN; FENG, 2001).

Em problemas dinâmicos, onde há uma alta mobilidade de partículas, há a necessidade que o balanceamento de carga seja realizado em conformidade com a movimentação das partículas. Isto ocorre sobretudo nas técnicas geométricas de particionamento de domínio. Em geral, estas técnicas estão fundadas na definição de subdomínios espaciais e zonas de comunicação entre subdomínios. Quando uma partícula desloca-se além do subdomínio espacial de um processo, a mesma deve deixar de existir naquele processo para surgir em outro. Este procedimento pode acarretar desbalanços de carga, já que um processo terá mais partículas e portanto mais trabalho que o outro.

Nas técnicas de particionamento baseadas em topologia, não existe uma definição de subdomínio espacial que force a migração das partículas. No entanto, novamente devido à mobilidade das partículas, o grafo de relações que indica a topologia pode ser alterado. Nesta nova configuração, o número de cortes de arestas para o particionamento adotado pode aumentar significativamente ao longo da simulação. Estas mudanças elevam o volume de comunicação entre processos e devem ser evitadas.

Deve-se salientar que para cada passo de tempo da simulação, cada processo contém partículas que pertencem ao subdomínio correspondente e cópias temporárias de partículas de processos vizinhos. Com base nestes volumes de partículas, é possível estabelecer uma métrica que reflete o grau de comunicação entre processos na forma:

$$\delta_{imbal} = \frac{m \max(N_i^{(t)})}{\sum_{i=1}^{m} N_i} - 1, \tag{3.4}$$

onde N_i é o número de partículas do subdomínio i, m é o número de processos paralelos e $N_i^{(t)}$ é o número total de partículas no processo i, incluindo cópias de partículas vizinhas. Observe que, δ_{imbal} assume valores próximos de zero quando as comunicações entre processos são mínimas. Qualquer outro valor positivo representa um indicador de volume de comunicação que pode ser utilizado como referência de comparação.

Seja devido à migração de partículas entre processos ou decorrente do aumento no número de cortes de arestas de grafo, as estratégias de balanço de carga devem ser empregadas. Algumas estratégias simples de controle podem ser adotadas com este objetivo.

3.4.1 Controle por desbalanço de particionamento

Uma forma simples de controle de desbalanço de cargas em uma simulação do DEM consiste em checar a diferença percentual entre o valor máximo e o valor médio de partículas contidas nos subdomínios dos m processos envolvidos. O desbalanço em número de partículas δ_p pode ser escrito na forma:

$$\delta_p = \frac{\max(N_i)}{\frac{1}{m}\sum_{i=1}^{m} N_i} - 1, \tag{3.5}$$

onde N_i indica o número de partículas para o i-ésimo processo. Valores de δ_p próximos de zero indicam um bom balanço de carga. Sugere-se que um novo particionamento seja realizado quando $\delta_p > 0,1$.

Embora o desbalanço em número de partículas seja uma boa métrica de controle para particionamentos geométricos, o mesmo não é bem aplicado em técnicas de particionamento onde o número de partículas nos processos não varie ao longo da simulação.

3.4.2 Controle por deslocamento limite

Na estratégia de balanço de carga baseada em deslocamento limite considera-se o valor do deslocamento máximo possível em um histórico recente como critério de controle para balanço de carga. Esta técnica não se baseia na migração de partículas entre processos, funcionando inclusive nos casos onde o número de partículas do subdomínio não muda.

O procedimento consiste em estimar o máximo deslocamento que uma partícula hipotética poderia assumir em um histórico recente. Adota-se que esta partícula possui velocidade igual à máxima velocidade, em módulo, dentre todas as partículas do domínio.

Desta forma, o deslocamento recente da partícula hipotética (d_{hr}) pode ser estimado através de uma estratégia de atualização em marcha. Para cada instante de tempo de simulação, atualiza-se d_{hr} através da expressão:

$$d_{hr} \leftarrow d_{hr} + \max(|v_i|)\Delta t, \tag{3.6}$$

onde $\max(|v_i|)$ é a maior velocidade em módulo para as partículas do domínio no instante corrente. O incremento de tempo de integração utilizado na simulação é representado por Δt.

Note que o deslocamento d_{hr} cresce tão rápido quanto maior for a mobilidade das partículas. Pode-se portanto definir um valor limite para d_{hr} a partir do qual realize-se o balanceamento de carga. Este valor limite, denotado por $d_{\max}$, tem influência, portanto, na frequência de re-particionamento de domínio. Quanto menor seu valor, mais frequentes serão os re-particionamentos indicados.

Alternativamente, pode-se expressar $d_{\max}$ como função do raio das partículas. Sendo r_i os raios das partículas do modelo, pode-se definir o deslocamento limite como:

$$d_{\max} = \lambda \max(r_i), \tag{3.7}$$

onde λ o parâmetro de controle proporcional ao raio das partículas.

A estratégia de atualização do deslocamento d_{hr} repete-se durante toda a simulação. Sempre que o deslocamento atingir o valor limite especificado, sugere-se que seu valor seja reinicializado com zero. Ou seja:

$$d_{hr} \leftarrow 0 \; se \; d_{hr} > d_{\max}. \tag{3.8}$$

Esta técnica de controle de balanceamento é particularmente útil em estratégias de particionamento topológicas. Nestes métodos de decomposição de domínio, o número de partículas de cada subdomínio é mantido constante enquanto não ocorrem re-particionamentos. Entretanto, sabe-se que o número cópias de partículas vizinhas altera dinamicamente durante os passos de tempo intermediários.

3.4.3 Técnicas de balanceamento de carga

Conforme comentado anteriormente, as técnicas de particionamento tem como objetivo a redução do número de cortes de arestas e o balanceamento de cargas entre subdomínios. Para atingir tais propósitos, em geral não é feita a consideração sobre o volume de comunicação necessário quando o subdomínio muda de uma configuração para outra. Estas mudanças podem, por exemplo, ser decorrentes do movimento de nós ou refinamento de malhas de elementos finitos ou da cinemática de partículas em modelos descontínuos. Esta seção apresenta alguns procedimentos utilizados para a estratégia de balanceamento dinâmico de carga.

3.4.3.1 Balanceamento através de re-particionamento

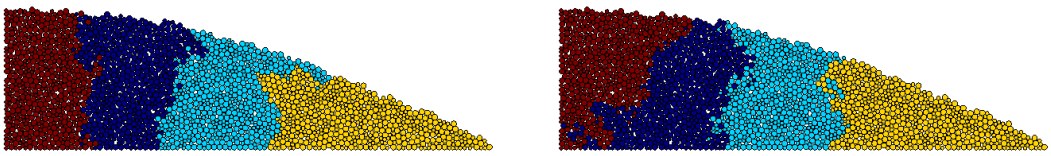
As estratégias de balanceamento dinâmico por re-particionamento são de aplicação relativamente direta. Qualquer técnica de particionamento disponível pode ser empregada visando o balanceamento. Em alguns casos, no entanto, é necessário um tratamento dos dados para que os particionamentos consecutivos que são gerados não sejam bastante sensíveis a pequenas perturbações. Mudanças bruscas de particionamento acarretam grandes volumes de comunicações e portanto devem ser evitadas no processo de balanceamento.

As estratégias de re-particionamento geométrico são extensivamente estudadas, uma vez que são menos sensíveis a perturbações. O motivo é que estes procedimentos são determinísticos, ou seja, múltiplos particionamentos de uma mesma configuração geram resultados idênticos. No entanto, a desvantagem de uso destas técnicas é que elas tendem a produzir decomposições de piores qualidades.

Embora o particionamento de domínio baseado em topologia conduza a boas qualidades de decomposições, deve-se atentar para as implicações de particionamentos sucessivos necessários quando do seu uso em estratégias de paralelização. Em geral, problemas dinâmicos exigem que a tarefa de particionamento seja realizada consecutivamente ao longo do tempo ou mesmo de iterações. O que se observa no entanto é que, mesmo para pequenas mudanças no arranjo da topologia, se a tarefa de decomposição for realizada de forma completa (independentemente de particionamentos anteriores), os subdomínios gerados em particionamentos consecutivos são bastante distintos. Isso acaba, muitas vezes, aumentando o volume de dados que são trocados entre processos e diminuindo a eficiência da paralelização.

A Figura 27 ilustra o problema do particionamento consecutivo realizado de forma independente. Para tanto, apresentam-se em detalhe dois particionamentos correspondentes aos instantes de tempo consecutivos, sendo Δt o incremento de tempo de integração usado na solução do método numérico adotado.

Figura 27 – Detalhe de particionamento consecutivo (movimento).



(a) Instante t (b) Instante $t + \Delta t$

Observa-se que a fronteira que delimita geometricamente o subdomínio de um dos processos varia substancialmente para pequenas variações da topologia.

Diversas técnicas de balanceamento dinâmico tem sido utilizadas para modificar particionamentos de grafos (DRIESSCHE; ROOSE, 1995). Uma destas técnicas baseia-se na criação de vértices virtuais para diminuir o volume de migração de nós entre subdomínios. Dado um conjunto de subdomínios gerados por um particionamento prévio, criam-se vértices virtuais para cada um dos subdomínios. Todos os vértices dos subdomínios são, desta forma, ligados aos correspondentes vértices virtuais criados. Uma função de peso pode ser atribuída às arestas virtuais de tal maneira que reflita o custo de comunicação entre processos.

Outra estratégia para balanceamento de carga por re-particionamento de grafos utiliza um conceito de correspondência local para a definição dos subdomínios e dos volumes a serem migrados. Esta estratégia é especialmente interessante no uso de algoritmos de particionamento multi-nível (SCHLOEGEL; KARYPIS; KUMAR, 2001).

3.4.3.2 Balanceamento através da migração de nós

Outra estratégia de balanceamento de carga entre subdomínios consiste no procedimento de migração de nós do contorno. Esta técnica é, em geral, mais complicada que as metodologias baseadas em re-particionamentos e envolve basicamente duas etapas:

- Cálculo de fluxo: Procedimento responsável pela determinação do volume de vértices do grafo a serem migrados entre subdomínios para atingir o balanceamento de carga;
- Seleção de nós: Etapa que decide quais dos nós dos subdomínios devem ser migrados.

O emprego das técnicas de balanceamento por migração de nós exige ainda, quando no uso em aplicações paralelas, a tarefa de migração de nós entre subdomínios. Esta etapa é complexa e depende da estrutura de dados empregadas nas aplicações. No caso de aplicações de elementos discretos, por exemplo, este procedimento deve transferir, de um subdomínio (processo) para outro, os dados referentes às partículas do modelo.

O problema de cálculo de fluxo pode ser definido como um procedimento de determinação de transferências de cargas entre subdomínios. Para um dado grafo $G = (V, E)$, formado por um conjunto de vértices V e arestas E, associa-se a cada vértice $i \in V$ uma carga l_i associada. O grafo aqui apresentado representa a topologia dos subdomínios, onde os vértices representam os subdomínios, e as arestas indicam os contornos compartilhados entre subdomínios. O cálculo de fluxo objetiva a determinação dos volumes transferidos entre subdomínios incidentes, dado por δ_{ji}, de tal forma que:

$$l_i + \sum_{j \leftrightarrow i} \delta_{ji} = \bar{l},\ i \in V, \tag{3.9}$$

onde $\bar{l}$ representa a carga média, definida por:

$$\bar{l} = \frac{\sum_{i \in V} l_i}{|V|}. \tag{3.10}$$

Na determinação dos volumes δ_{ji} consideram-se as arestas do grafo que ligam os subdomínios i e j, representado pelo conjunto $j \leftrightarrow i$. Trata-se de um procedimento de solução de sistema de equações lineares, onde nem sempre é possível ter uma resposta única. Isso ocorre, uma vez que no grafo que representa a topologia do particionamento, em geral, existem mais arestas que vértices. Logo, o número de incógnitas $|E|$ é maior que o número de equações $|V|$. As soluções mais comuns para o cálculo de fluxo baseiam-se em métodos baseados em difusão, algoritmos multi-nível, métodos potenciais e algoritmos baseados em programação linear.

Na etapa de seleções de nós, deve-se escolher nós dos contornos dos subdomínios para transferência entre os subdomínios em questão. Este procedimento deve satisfazer o volume de fluxo previamente calculado.

Diversas técnicas podem ser empregadas para a seleção de nós para a migração. No trabalho de Walshaw, Cross e Everett (1997) o conceito de ganho relativo é empregado para a seleção dos nós. Já no trabalho de Schloegel, Karypis e Kumar (1997), o algoritmo paralelo do METIS foi estendido para incorporar o recurso de balanceamento dinâmico da carga. A técnica empregada baseia-se em duas etapas: difusão multi-nível e refinamento multi-nível. Por fim, no trabalho de Ou e Ranka (1997), utiliza-se um procedimento de seleção de nós que é baseado na partição com o maior número de nós adjacentes.

Deve-se ter em mente que o procedimento de seleção de nós deve ser realizado de forma a minimizar o número de cortes de arestas entre subdomínios e a migração de dados.

4 Sistema computacional e otimizações do código serial

As técnicas de paralelização, quando realizadas de forma apropriada, contribuem na diminuição do tempo de processamento dos mais diversos modelos numéricos. Isso acontece uma vez que tarefas caras computacionalmente são fragmentadas em outras menores que são processadas de forma simultânea, e portanto mais rápida.

Um fator importante que influencia a performance de programas paralelos com memória distribuída é a forma como a decomposição de domínio é realizada. Esta tarefa tem implicações diretas, tanto no balanço de carga, como também nas comunicações necessárias entre processos. Deve-se ter em mente que as comunicações em sistemas distribuídos requer o uso de mecanismos de troca de mensagens e de dispositivos físicos de comunicação entre processos. Em um *cluster* de computadores, esta comunicação possui velocidade limitada pelos recursos de comunicação em rede disponíveis.

Além dos recursos de paralelização, uma boa melhoria de performance pode ser obtida através de otimizações de códigos seriais. Em geral, os códigos que implementam rotinas paralelas devem ser baseados em códigos seriais bem implementados, o que exige do programador uma boa experiência com o problema simulado e habilidade de programação. Isso envolve, dentre outros fatores, a escolha de algoritmos, estruturas de dados, implementações mais apropriadas para o *hardware* utilizado e o melhor uso da memória.

As otimizações realizadas em códigos seriais potencializam o ganho de desempenho em execuções de códigos paralelos. Pode-se imaginar, de forma simplificada, que o tempo demandado em uma execução de código paralelo é composto por duas frações distintas. A primeira fração é aquela que contém unicamente o tempo de comunicação entre processos e a segunda é aquela que contém, exclusivamente, o processamento, idealizado como serial em cada um dos processos. Note que a última fração é aquela que tende a ser reduzida com o aumento do número de processos. Já a primeira, tende a aumentar uma vez que mais interfaces entre processos são criadas.

Assim sendo, uma das maneiras de otimizar um código paralelo consiste em melhorar sua porção serial. O estudo de *profile* é a forma usual de avaliar o código de um sistema computacional. Esta técnica quantifica o tempo computacional demandado em cada porção de uma rotina. A medição de tempo pode ser feita para cada função ou método do código, ou ainda considerando cada linha de sua implementação. As implementações realizadas neste trabalho utilizam o sistema computacional DEMOOP, apresentado na seção 4.1. O respectivo estudo de *profile* realizado para o *software* é discutido em maiores detalhes na

seção 4.2.

Outra questão com bastante influência no tempo de processamento computacional é a forma de acesso aos dados na memória. Isto acontece uma vez que os processadores modernos comumente empregados em ambientes de computação utilizam estratégias hierárquicas de acesso à memória. Uma melhor performance computacional é obtida quando a rotina que implementa uma dada tarefa toma proveito destas estratégias. Códigos científicos normalmente não tomam cuidado suficiente com os modelos de acesso à memória, mas a simulação de modelos de larga escala pode exigir otimizações neste nível. A seção 4.3 apresenta algumas técnicas de otimização de acesso à memória que podem ser aplicadas em simulações de elementos discretos.

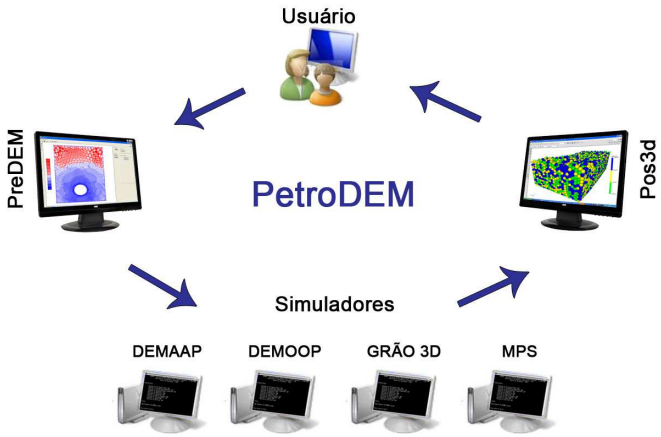
4.1 Sistema computacional DEMOOP

O DEMOOP (*Discrete Element Method Object Oriented Programming*) é um sistema orientado a objetos responsável pela análise de sistemas particulados, através do Método dos Elementos Discretos. Trata-se de uma ferramenta computacional que permite a simulação de meios discretos através da utilização de partículas. Estas entidades interagem entre si, tanto no espaço como no tempo, através de modelos de contato diversos. Através de um processo de integração numérica temporal é possível simular o comportamento destas entidades e investigar a dinâmica deste sistema idealizado. Associações podem então ser feitas para que o sistema analisado represente fenômenos físicos reais, dando suporte às mais diversas análises de engenharia. Detalhes sobre a implementação computacional do *software*, incluindo sua hierarquia de classes e especificações para funcionamento, estão disponíveis no trabalho de Júnior, Cintra e Silveira (2006).

O DEMOOP é um módulo computacional que faz parte do PetroDEM. O objetivo do PetroDEM é unificar, em um sistema computacional integrado, o conhecimento espalhado em diversas universidades brasileiras relacionado ao Método das Partículas (PETRODEM, 2013). Este sistema, desenvolvido desde 2008, incorpora módulos computacionais avançados que permitem a fácil manipulação do grande volume de informação decorrente das análises numéricas de partículas. Além disso, o sistema foi projetado para lidar com elevadas demandas por processamento numérico de dados e técnicas avançadas de computação gráfica. Esta elevada demanda por recursos computacionais é um dos principais desafios motivadores de pesquisa para o método numérico atualmente.

O *workflow* básico de trabalho do PetroDEM é descrito na Figura 28. A comunicação entre os módulos do sistema é realizada através de arquivos no formato neutro (MENEZES; CARVALHO; MIRANDA, 2002). A interface do sistema com o usuário é feita de forma interativa através de um pré-processador de dados (PreDEM) (FERRAZ et al., 2008) e de um pós-processador de resultados (Pos3d) (CARVALHO; MARTHA; CELES, 1997).

Figura 28 – Workflow básico do PetroDEM.



Fonte: PetroDEM (PETRODEM, 2013).

Alternativamente, o pós-processamento de dados pode ser realizado através do *software* DEMView (AMORIM; CINTRA; LIRA, 2006), que permite a visão estereoscópica dos resultados em ambientes de projeção tridimensional, ou mesmo através do *software* ParaView (AHRENS; GEVECI; LAW, 2005b).

A saída de dados de simulação do DEMOOP pode ocorrer de forma paralela, utilizando arquivos de dados binários e arquivos de indexação. Atualmente a saída de dados é compatível com o *software* DEMView, em formato proprietário, e com o ParaView, em formato XDMF (XDMF..., 2013).

4.2 *Profile* de código

Visando avaliar a implementação computacional do código serial do DEMOOP, realizou-se o *profile* do seu código. O modelo numérico empregado no estudo foi idealizado de forma a reproduzir uma configuração usual de uma simulação por elementos discretos. Trata-se de um domínio cúbico de aresta unitária, composto por partículas de mesmo raio em um arranjo BCC (*Body Centered Cubic*) com posições perturbadas aleatoriamente. A perturbação adicionada ao arranjo foi imposta de maneira a induzir o cálculo de forças de contato, uma das principais tarefas do algoritmo do Método dos Elementos Discretos. A Figura 29 ilustra a disposição inicial das partículas.

Para o estudo de *profile* realizam-se simulações numéricas envolvendo modelos discretos contendo de 250 a 2,5 milhões de partículas. Os modelos são simulados durante um total 100 passos de solução numérica, executando de forma serial. Um resumo dos resultados de *profile* é apresentado na Tabela 1. A ferramenta de análise de código empregada nos testes foi o *gprof* (GNU Profiler) (GRAHAM; KESSLER; MCKUSICK, 2004).

É possível notar em todos os casos da Tabela 1, que a tarefa com mais representação em termos de tempo de processamento é a de cálculo de forças internas. De fato ela constitui

Figura 29 – Arranjo BCC perturbado (2546 partículas).

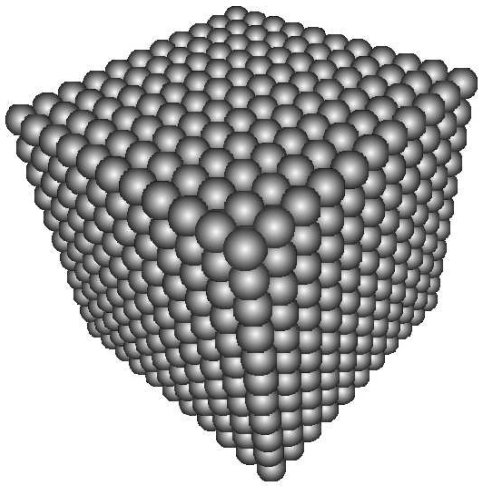


Tabela 1 – *Profile* do código serial do DEMOOP (Busca baseada em *hash table*).

Número de Partículas:	**250**	**2,5k**	**25k**	**250k**	**2,5M**
1 - Cálculo de Força Interna	99,01	96,90	95,71	96,78	94,63
1.1 - Busca de contatos (Mapeamento Direto)	10,00	6,38	6,56	4,51	5,21
1.2 - Inicialização do teste de contato	30,00	15,6	9,25	6,63	1,99
1.3 - Cálculo da Força de Contato	59,56	70,64	78,95	85,30	86,00
1.3.1 - Escolha do teste de intersecção	10,00	4,26	1,80	1,21	1,35
1.3.2 - Teste de intersecção geométrica	30,00	24,11	21,22	19,91	13,89
1.3.3 - Leitura da estrut. de dados de contato	-	3,55	13,22	13,99	22,49
1.3.4 - Gravação na estrut. de dados de contato	-	17,02	16,23	30,27	30,12

Nota: Dados apresentados em valores percentuais do tempo total de execução. O símbolo – indica valores desprezíveis.

a tarefa básica do Método dos Elementos Discretos, que modela numericamente o fenômeno físico de contato. Essa tarefa engloba uma série de outras ações, dentre as quais: busca de contatos, teste de intersecção geométrica e cálculo de forças de contato.

Observa-se ainda do *profile* uma diferença de comportamento para os diferentes modelos numéricos. Note que à medida que o modelo numérico cresce (aumento do número de partículas) a relevância das tarefas de leitura e gravação na estrutura de dados de contato aumenta. Para o caso do modelo envolvendo $2,5$ milhões de partículas, por exemplo, estas tarefas correspondem a mais de 52% do tempo total de execução da simulação.

A estrutura de dados de contatos é responsável por armazenar informações passadas relativas ao contato de pares de partículas. A informação contida nessa estrutura serve para o cálculo de forças em alguns modelos de contato, a exemplo daqueles que representam o fenômeno de atrito. Na forma até então implementada, o acesso a esses dados é feito através de uma estrutura de dados baseada em *hash table*. Essa alternativa é válida para pequenos modelos, conforme já observado na Tabela 1, no entanto tem oferecido restrições de performance para modelos de maiores escala. Assumindo que se deseje simular modelos complexos envolvendo centenas de milhões de partículas, o uso dessa estrutura de dados

torna-se impraticável.

Visando otimizar o código computacional do sistema anteriormente apresentado, uma implementação que altera a forma atual de acesso à estrutura de dados de contatos foi realizada. Na proposta da implementação os dados de contatos são armazenados em matrizes bidimensionais esparsas e simétricas, onde os índices de acesso correspondem aos identificadores da partícula (valor numérico inteiro positivo menor ou igual ao número de partículas do modelo). Embora essa forma de acesso envolva uma busca para resgatar informações de um dado par, ela é realizada em um domínio menor do que aquele praticado quando no uso da estrutura *hash table*. Isso acaba contribuindo positivamente em tempo de processamento, uma fez que o tempo de busca deixa de crescer com o tamanho do modelo.

Visando avaliar a mudança de desempenho do sistema computacional decorrente da utilização de matriz esparsa para armazenamento de contatos, apresenta-se a seguir um estudo comparativo de *profile* e performance. Os dados obtidos são comparados com os correspondentes obtidos pela versão do sistema que utiliza a estrutura *hash table* para armazenar informações de contato entre partículas. O *profile* realizado assume a simulação serial do modelo de $2,5$ milhões de partículas com arranjo BCC perturbado anteriormente apresentado. A Tabela 2 resume os percentuais de tempo gastos nas tarefas de maior custo computacional para as versões.

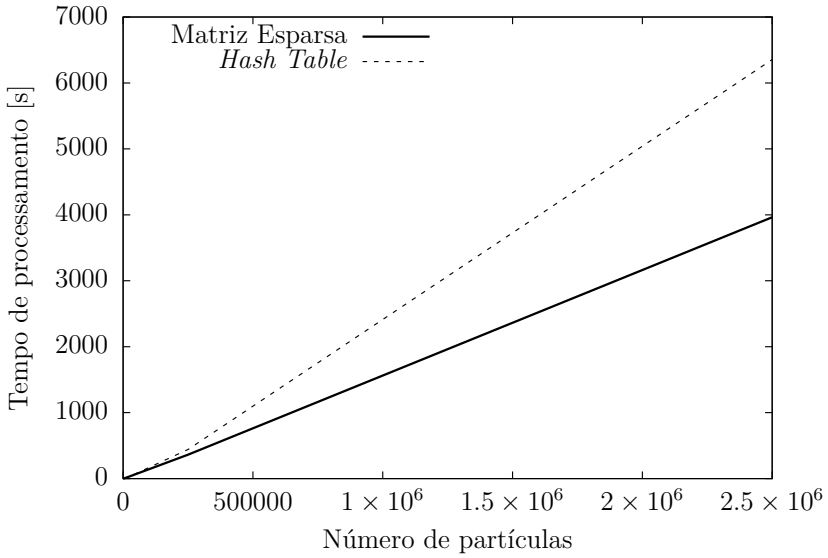
Tabela 2 – Comparativo de *profile* do código serial do DEMOOP considerando diferentes estruturas de dados.

Modelo de armazenamento:	***Hash Table***	**Matriz Esparsa**
1 - Cálculo de Força Interna	94,63	92,69
1.1 - Busca de contatos (Mapeamento Direto)	5,21	8,84
1.2 - Inicializa teste de contato	1,99	3,11
1.3 - Cálculo da força de contato	86,00	78,81
1.3.1 - Escolha do teste de intersecção	1,35	2,03
1.3.2 - Teste de intersecção geométrica	13,89	28,34
1.3.3 - Leitura da estrut. de dados de contato	22,49	10,88
1.3.4 - Gravação na estrut. de dados de contato	30,12	-
1.3.5 - Consulta de dados do modelo	7,40	15,77

Nota: Dados apresentados em valores percentuais do tempo total de execução. O símbolo – indica valores desprezíveis.

Note que, na versão baseada em matriz esparsa, o acesso à estrutura de dados de contato demanda bem menos tempo de processamento que na anteriormente praticada. Devido à esta redução, as frações de tempo de processamento demandado nas demais tarefas acabam sendo alteradas. Evidencia-se portanto, nesta nova configuração, o teste de intersecção geométrica (tarefa bastante executada durante o processo de determinação de forças de contato, mas de baixa complexidade de implementação). A Figura 30 confronta as implementações em termos de tempo de processamento computacional.

Figura 30 – Comparação de desempenho das versões.



Conforme observa-se, a implementação baseada em matriz esparsa tem um desempenho computacional significativamente melhor, sobretudo nos modelos de maior escala. Isso acontece em virtude da diminuição do tempo de acesso à estrutura de dados de contato. Note ainda o comportamento linear para a relação entre número de partículas e tempo de processamento. Este fato é esperado em virtude da escolha dos algoritmos utilizados na definição do modelo empregado no estudo de *profile*.

4.3 Otimização de acesso à memória

As relações de vizinhança em simulações do DEM são, geralmente, não estruturadas. Este fato é resultante da constante mobilidade das partículas e do aparecimento e extinção de contatos durante toda a simulação. Esta característica resulta em um padrão não regular de acesso à memória. Os acessos à memória em simulações do DEM são necessários para a leitura e atualização de informações tais como posições, velocidades, geometrias, dentre outras. Seus processamentos relacionados causam uma má utilização da CPU se nenhum cuidado especial for tomado. Isto deve-se basicamente à velocidade de acesso aos dados alocados em memória principal (RAM - *Random Access Memory*).

Os processadores atuais utilizam estratégias hierárquicas de acesso à memória no intuito de tentar reduzir o tempo médio de consulta a dados. Para tanto, os mesmos dispõem de dispositivos auxiliares que permitem operações de cópia temporária de dados, as chamadas memórias *cache*. Estas memórias, por motivos tecnológicos e econômicos, possuem capacidades de armazenamento reduzidas mas operam em velocidades mais altas de acesso. Dependendo do tamanho de um modelo computacional, toda a memória demandada para armazenar as estruturas de dados envolvidas pode caber na memória

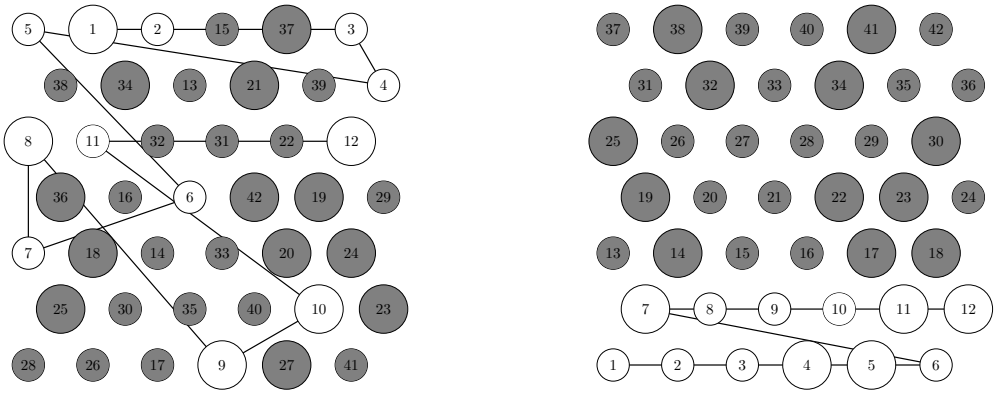
cache. Entretanto, modelos de larga escala costumam extrapolar bastante seus limites de capacidade.

O computador utiliza as memórias *cache* para evitar o acesso contínuo e dispendioso à memória principal (RAM -). O procedimento realizado consiste em copiar porções da RAM para a memória *cache*, e realizar as operações de cálculo utilizando as cópias temporárias de maior velocidade de acesso. Quando uma operação de cálculo demanda o uso de um dado que não está presente na memória *cache*, um novo acesso à memória principal é realizado. Este evento é chamado de *cache-miss*. Em geral, quanto menor o número de eventos de *cache-miss* mais rápido acontece o processamento dos dados.

Quando um código computacional é consciente do uso da memória *cache*, o mesmo tenta explorar a localidade dos dados para tomar vantagem da melhor performance de cálculo provida pela hierarquia de *cache*. O padrão de acesso à memória em simulações do DEM é potencialmente definido pelo procedimento de busca de contatos. Esta tarefa visa encontrar pares de objetos, partículas ou obstáculos, que são espacialmente próximos. Quando existem diferenças significativas entre a localidade espacial dos objetos e a localidade dos dados, os eventos de *cache-miss* são mais comuns no processamento de um dado conjunto de partículas. Conjuntos de partículas envolvendo um número elevado de elementos também evidenciam a influência dos eventos de *cache-miss* no tempo de processamento.

A localidade dos dados em sistemas particulados é basicamente definida pela posição da partícula na memória. Geralmente, a alocação destes elementos é determinada pela sequência inicial fornecida pelas rotinas de geração de partículas. Estes procedimentos de geração podem criar sequências arbitrárias de partículas, desde o arranjo lexicográfico (HAN; TSENG, 2006) geralmente presente em estruturas cristalinas como o BCC (*Body-Centered Cubic*), até arranjos completamente aleatórios (FRERY et al., 2012). A Figura 31 ilustra estes dois cenários.

Figura 31 – Localidade dos dados em sequências de partículas.



(a) Sequência aleatória.

(b) Sequência lexicográfica.

Note que a sequência aleatória não assegura localidade dos dados por natureza. A sequência lexicográfica, em contraste, apresenta uma melhor localidade dos dados uma vez que seleciona os vizinhos em um direção coordenada. O arranjo lexicográfico é uma generalização da ordem dada às palavras em um dicionário. No caso em questão, Figura 31, as "palavras"podem ser idealizadas como as posições espaciais das partículas e as "letras"representam as direções coordenadas. Diversas técnicas de ordenamento de memória tentam otimizar o uso da hierarquia *cache* através de procedimentos auxiliares à simulação. Fleissner e Eberhard (2005) apresentam um mecanismo de agrupamento que emprega caixas limite visando utilizar melhor o mecanismo de *cache* em modelo do DEM. O método no entanto requer conhecimento de *hardware* com o objetivo de definição do tamanho destas caixas. As seções seguintes descrevem os métodos de otimização de acesso à memória utilizados neste trabalho.

4.3.1 Partitioned Spatial Sorting

O método PSS (Partitioned Spatial Sorting), introduzido por Berger et al. (2015), é uma adaptação do algoritmo de ordenamento espacial original apresentado por Yao et al. (2004). Este procedimento realiza duas ações principais: agrupamento por partição e ordenação por posição espacial. Seu desenvolvimento foi motivado pelo uso em sistemas de paralelização híbridos que requer, além do ordenamento de memória, um procedimento de agrupamento de partículas.

A divisão de partições no método pode usar um critério arbitrário. Entretanto, Berger et al. utilizam um método de particionamento local baseado no algoritmo RCB, conforme apresentado na subseção 3.1.2. Os autores também consideram uma estratégia de balanço de carga baseado no número de contatos das partículas. O procedimento de particionamento local inicia com a geração de uma etiqueta para cada partícula. Estas etiquetas definem as partições locais e servem para agrupar o vetor de partículas em blocos.

Depois do agrupamento de partículas por etiqueta, a ordenação por posições espaciais é então realizada para cada bloco de partículas. Dentro de cada bloco, este ordenamento segue o algoritmo original proposto por Yao et al. (2004). A ordenação espacial considera a caixa limite que contém as partículas e seleciona a direção mais alongada dentre as direções coordenadas. Em seguida, as partículas são ordenadas em memória considerando as projeções de suas posições na direção mais alongada. Partículas que estão espacialmente próximas tendem a ser alocadas próximas no vetor de partículas, aumentando desta forma a performance de uso da memória *cache*.

Embora o método PSS tenha sido desenvolvido para funcionar de forma particionada através de *threads*, tanto este método quanto o método de ordenação espacial original podem ser usados para otimizar o acesso à memória em simulações do DEM.

4.3.2 HSFC

No método baseado na curva de preenchimento de espaço de Hilbert (HSFC), as direções coordenadas não são utilizadas preferencialmente no processo de ordenação em memória. Ao invés disto, a curva de Hilbert é adotadas como referência. Neste método, cada partícula pode ser projetada em um espaço normalizado, a partir de suas coordenadas e da caixa limite envolvente do conjunto, e então mapeada em um espaço unidimensional. Na sequência, a estratégia de otimização de memória ordena o vetor de partículas usando as projeções obtidas para o conjunto. Este procedimento melhora a performance de acesso à memória por conta da propriedade de localidade da curva de Hilbert. Rotinas que mapeiam uma coordenada tridimensional no espaço unitário projetado de Hilbert podem ser encontradas na biblioteca Zoltan (BOMAN et al., 2012).

A curva de Hilbert é um fractal contínuo descrito por Hilbert (1891). Conforme comentado, assim como outras funções de preenchimento de espaço, o domínio da curva de Hilbert é unitário. De uma forma mais geral, a curva de Hilbert é uma função de mapeamento que têm como domínio os hipercubos unitários n-dimensionais. Trabalhos anteriores demonstraram que a curva de Hilbert atinge melhor grau de agrupamento que outras curvas de preenchimento de espaço tais como a *z-curve* e a *gray-coded* (MOON et al., 2001). A propriedade de agrupamento indica que a localidade entre objetos no espaço multi-dimensional é preservada no espaço linear.

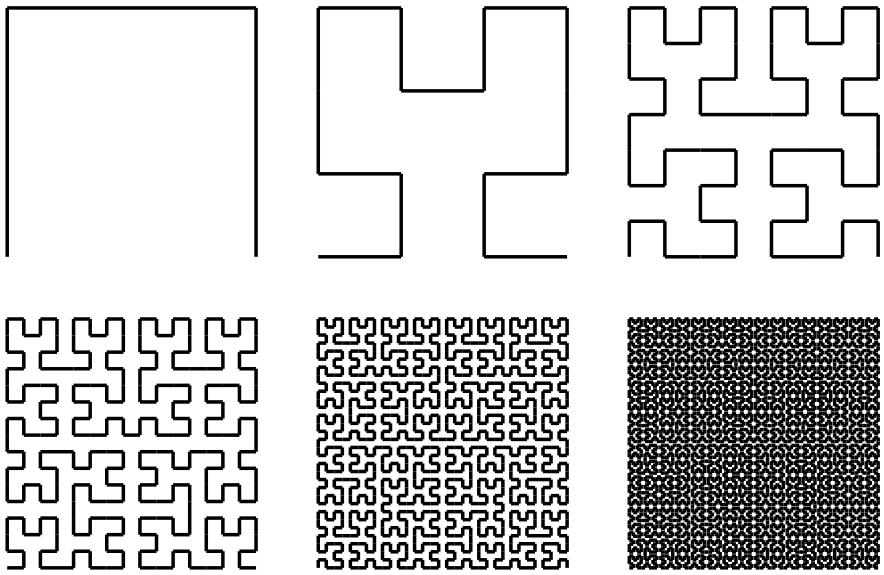
A construção da curva de Hilbert pode ser idealizada como um processo recursivo que cria copos (quadrados com um lado aberto) e juntas (linhas que conectam dois copos). Este procedimento inicia com um único copo, na curva de primeira ordem de Hilbert. Na segunda etapa, o copo inicial cria quatro copos e três juntas. Em cada ordem crescente, o processo de replicação dos copos e juntas repete-se, conforme ilustrado na Figura 32. Este processo pode repetir infinitamente e a cada etapa de replicação o domínio vai sendo preenchido cada vez mais.

A ordenação das partículas usando HSFC acontece a partir do mapeamento de um hipercubo n-dimensional para o intervalo unitário. Especificamente, o mapeamento é realizado a partir de um quadrado ou cubo unitário para o intervalo linear unitário ($[0, 1]$), nos casos bidimensional e tridimensional respectivamente. As funções de mapeamento utilizam a estratégia proposta por Lawder e King (2001), cuja implementação computacional é provida pela biblioteca Zoltan. Visando assegurar que o domínio de mapeamento é um quadrado ou cubo unitário, as posições espaciais das partículas ($\boldsymbol{x}$) são normalizadas pelas dimensões da caixa limite envolvente do conjunto. Desta forma, as coordenadas normalizadas são definida por:

$$x_{ik}^{norm} = \frac{x_{ik} - \min(x_{ik})}{\max(x_{ik}) - \min(x_{ik})}, \tag{4.1}$$

onde i denota o índice da partícula e k é a direção coordenada ($k = 1...dim$).

Figura 32 – Construção da curva de Hilbert por iterações.

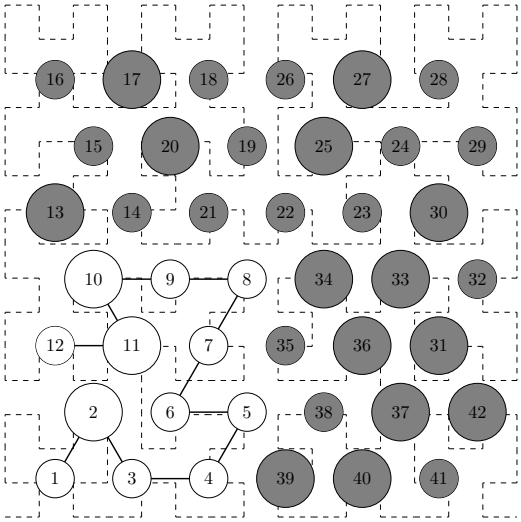


O arranjo de partículas baseado em HSFC depende da projeção de cada partícula no espaço unitário. Para a i-ésima partícula, esta projeção pode ser representada por hsfc_i. O procedimento de ordenação de memória consiste em ordenar o vetor hsfc. Este procedimento deve utilizar um vetor auxiliar contendo os índices das partículas para permitir o correto mapeamento de cada um destes elementos na nova alocação. Depois deste processo, cada partícula recebe um número de sequência. Os números de sequência sugerem como alocar as partículas na memória, aumentando de acordo com a projeção na curva de Hilbert. A Figura 33 ilustra os números de sequência, projetados para cada partícula, e a curva de Hilbert de referência, apresentada em linha pontilhada desde seu início (canto inferior esquerdo) até seu fim (canto inferior direito).

Note que a localidade dos dados, representada pelos números de sequência, apresenta uma boa correlação com a localidade espacial. Observa-se que partículas com números de sequência próximos estão, em grande parte, localizadas espacialmente próximas também de acordo com este procedimento de ordenação.

Problemas dinâmicos e com alta mobilidade de partículas exigem que o procedimento de ordenação seja repetido durante o processo de simulação. Uma estratégia de controle pode, no entanto, ser empregada com o intuito de sugerir a frequência de ordenação de memória. Assim como nas técnicas de balanceamento de carga, o critério de controle por deslocamento limite pode ser adotado com esta finalidade (subseção 3.4.2). Este indicador reflete o grau de mobilidade das partículas, sendo portanto bastante aplicável. Deve-se ter em mente que as interações entre partículas alteram com uma frequência diretamente proporcional à sua mobilidade. Neste trabalho, o deslocamento limite adotado

Figura 33 – Projeção de um conjunto de partículas na curva de Hilbert.



na estratégia de controle é de duas vezes o raio máximo dentre as partículas do modelo ($\lambda = 2$). Este valor representa, geometricamente, o diâmetro de uma partícula. É bastante provável portanto que deslocamentos desta magnitude sejam suficientes para que espaços vazios sejam criados e preenchidos, alterando significativamente o mapa de interações entre elementos.

5 Estratégias de paralelização distribuída

Problemas de larga escala no DEM podem surgir de discretizações refinadas para ambos os domínios, espacial e temporal. Quando a solução espacial do domínio é um fator importante para representar um fenômeno físico ou um especificação geométrica, o conjunto de partículas que representa este domínio deve crescer. Simulações numéricas que lidam com grandes conjuntos de partículas podem ter soluções inviáveis devido ao tempo de processamento excessivo ou outras restrições de recursos computacionais, tais como o uso excessivo de memória. A computação paralela é, portanto, uma estratégia que pode ser utilizada para tornar estas simulações viáveis.

5.1 Paralelização distribuída

A formulação do DEM permite que seus elementos sejam resolvidos de forma independente, considerando apenas interações entre elementos localizados espacialmente próximos. Em decorrência deste fato, algumas estratégias de paralelização realizam uma divisão da análise do domínio espacial em subdomínios visando acelerar o processamento do método. Estes subdomínios são analisados por núcleos de processamento distintos quase que de forma independente. A comunicação entre os processos responsáveis pela análise de subdomínios ocorre basicamente devido a dois fatores: ocorrência de forças interativas entre partículas de subdomínios distintos (forças de contato ou coesivas) e deslocamento de partículas entre subdomínios. Em virtude da natureza do DEM, esta comunicação ocorre apenas entre as partículas localizadas nas regiões próximas às fronteiras de subdomínios.

Uma vez que a comunicação entre processos é um dos pontos cruciais para a eficiência da paralelização, devem-se desenvolver técnicas de decomposição de domínio e de balanceamento de carga, de tal maneira que a comunicação entre processos seja, sempre que possível, minimizada e que as tarefas dos processos sejam dinamicamente balanceadas ao longo da simulação. A comunicação pode, por exemplo, ser otimizada levando em consideração a arquitetura do ambiente de computação empregado. Os processadores atuais estão cada vez mais empregando arquiteturas *multi-core*. Isso, dentre outras implicações, permite que o fluxo de dados entre processos situados neste ambiente de memória compartilhada (CPU ou GPU) ocorra mais rápido do que aquele que utilizaria uma camada de rede para a troca de mensagens.

O processamento paralelo do método das partículas envolve ainda algumas questões relacionadas à própria entrada e saída de dados (i/o). Uma vez que grandes quantidades de dados são geradas ao longo das simulações, deve-se atentar à forma com que estas informações trafegam na rede ou mesmo como são armazenadas nos bancos de dados para

a etapa de pós-processamento. Estas tarefa deve ocorrer de maneira tal que não contribua negativamente para o tempo das simulações.

Para o projeto básico de um sistema paralelo de simulação de elementos discretos é necessário resolver alguns requisitos de funcionamento. Estes envolvem desde o mecanismo de geração do modelo, a sua simulação e consequentes atividades relacionadas, incluindo até a forma de visualização dos resultados. Dentre as principais etapas do processo destacam-se:

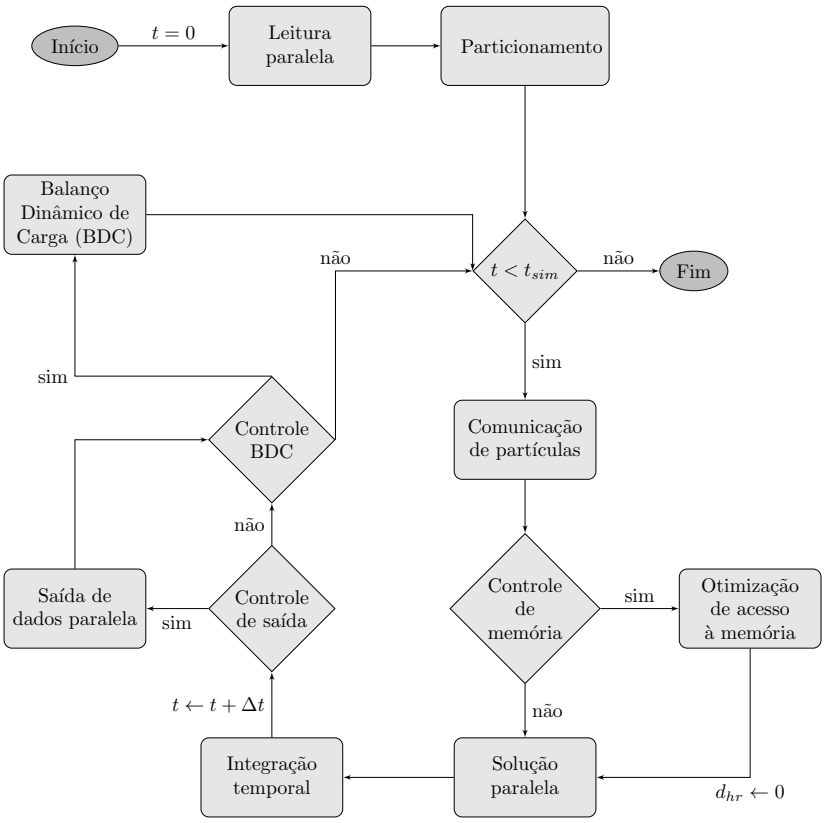
- Geração do domínio;
- Particionamento do domínio em subdomínios;
- Criação de processos com base nos subdomínios;
- Informação da topologia de comunicação entre processos;
- Definição de um modelo de atualização das interações entre sub-domínios;
- Comunicação entre subdomínios para compartilhar partículas de fronteira;
- Cálculo paralelo de forças interativas para as partículas do interior do subdomínio;
- Cálculo da contribuição de forças interativas para as partículas dos subdomínios vizinhos;
- Integração temporal;
- Balanceamento de carga e migração de dados;
- Geração de dados para pós-processamento paralelo;
- Pós-processamento paralelo.

5.2 Algoritmo geral

Neste trabalho propõe-se o fluxo de trabalho ilustrado na Figura 34 para atender as etapas e os requisitos apontados. O primeiro passo da metodologia é a leitura paralela do modelo. Esta tarefa visa carregar as informações de modelagem em cada processo, incluindo parâmetros do modelo de contato, as posições de todos os obstáculos e uma porção selecionada do conjunto de partículas. A leitura parcial das partículas é necessária porque o modelo completo pode ser muito grande para ser armazenado em um único processo. É altamente recomendado que a leitura inicial seja balanceada, ou seja, cada processo contém aproximadamente o mesmo número de partículas. Esta tarefa inicial entretanto não resulta ainda em uma partição de domínio computacional otimizada, mas evita que um determinado processo tente alocar memória excessiva. Em geral, um bom

particionamento de domínio é obtido quando o número de partículas em cada processo é uniforme e quando a comunicação requerida entre processos é minimizada.

Figura 34 – Metodologia de solução paralela distribuída para modelos do DEM.



Depois da leitura paralela do modelo, o procedimento segue para a tarefa de particionamento propriamente dita. Este mecanismo é necessário para distribuir as partículas entre os processos de acordo com uma distribuição que balanceia a carga de computação. A distribuição inicial fornecida pela leitura paralela é adotada como arbitrária na metodologia utilizada. Conforme mencionado anteriormente, o mecanismo de particionamento de domínio cria subdomínios para serem resolvidos em paralelo.

Uma vez que o procedimento de particionamento cria subdomínios balanceados, é necessário comunicar as partículas entre processos para calcular corretamente as forças de interação. Visando otimizar o acesso à memória e explorar a hierarquia *cache*, a tarefa de controle de memória é sugerida neste estágio.

Depois do controle de memória, o fluxo de trabalho segue para a solução paralela do modelo. Quando cada processo conhece seu conjunto atribuído de partículas e as partículas vizinhas do seu subdomínio, as forças de interação podem então ser calculadas para todos os subdomínios concorrentemente. O cálculo de forças deve ser realizado apenas para as partículas atribuídas ao subdomínio correspondente ao processo. As partículas vizinhas são necessárias apenas no intuito de transmitir forças de interação entre processos. Uma

vez que as forças de interação são calculadas, o conjunto de partículas vizinhas deve ser destruído.

A caixa limite que envolve o conjunto de partículas pode ser determinada visando melhorar o procedimento de busca de contatos envolvendo partículas e obstáculos. Desta forma, apenas os obstáculos que estão na caixa limite devem ser considerados na busca. Esta otimização reduz o número de testes geométricos desnecessários. O cálculo de forças de interação segue a formulação discutida na seção 2.6.

A evolução da cinemática do modelo obtida pelo DEM segue uma estratégia de marcha temporal que atualiza o tempo corrente t. Ela existe enquanto $t < t_{sim}$, onde t_{sim} é o intervalo de tempo de simulação. Portanto, o número de passos de análise é uma função que depende do tamanho do passo de tempo Δt. O'Sullivan e Bray (2004) apresentam uma metodologia para selecionar o tamanho do passo de tempo aplicável ao método de integração temporal utilizado no DEM.

Depois da atualização do instante de tempo corrente, a tarefa de controle de saída de dados é então conduzida. Este controle visa selecionar os passos de tempo que devem ser reportados para pós-processamento. A saída de dados do modelo na forma paralela é altamente recomendada para evitar um alto tráfego de informações através de troca de mensagens. Sistemas computacionais de visualização científica, tais como o ParaView (AHRENS; GEVECI; LAW, 2005a), possuem suporte para a leitura e renderização paralela de modelos distribuídos.

A tarefa final de cada passo de tempo é o controle do balanço de carga. Esta tarefa visa a otimização da eficiência paralela do sistema. As próximas seções deste capítulo discutem distintos estratégias de paralelização que adotam a metologia acima discutida. Estas estratégias apresentam propostas diferentes para os métodos de particionamento de domínio e balanço de carga. O emprego destas estratégias é melhor discutido no Capítulo 7, Capítulo 8, Capítulo 9 e Capítulo 10.

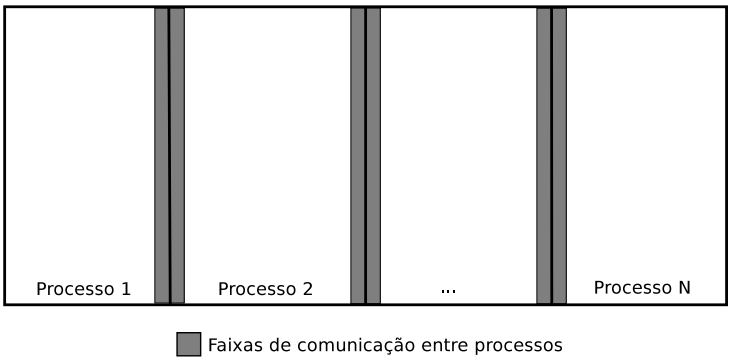
5.3 Paralelização baseada em faixas

A técnica de paralelização baseada em faixas é a estratégia mais simples de implementação paralela do DEM. Neste método o domínio é particionado através de faixas verticais ou horizontais. Estas faixas dividem o domínio a partir da informação da caixa limite global que contém todas as partículas. A largura das faixas pode ser uniforme ou considerar a distribuição espacial das partículas. Neste último caso a posição dos cortes pode ser escolhida de forma a tentar equilibrar a quantidade de partículas contida nas faixas.

Note que este método de decomposição de domínio resulta em uma topologia de

comunicação fixa, onde é possível determinar para cada processo quais são seus correspondentes vizinhos. Adota-se a estratégia de troca de mensagens entre processadores, seguindo o padrão MPI, e criam-se faixas limite de subdomínios onde as partículas são compartilhadas entre subdomínios vizinhos. A estratégia de comunicação adotada é a de transferência de partículas contidas nas faixas limite entre processadores responsáveis por subdomínios espaciais vizinhos. A largura destas faixas deve ser maior ou igual ao maior raio das partículas do domínio. A Figura 35 ilustra a estratégia de particionamento de domínio e as faixas de comunicação entre processos.

Figura 35 – Estratégia de particionamento de domínio por faixas.



Comunicações globais entre os processadores são necessárias, eventualmente, para o cálculo de propriedades globais, tais como na determinação da energia cinética do sistema (algoritmo de relaxação dinâmica).

O pré-processamento do sistema ocorre em duas etapas. Na primeira etapa o arquivo de dados (formato Neutral File/PetroDEM) é analisado com o intuito de definir a geometria inicial do modelo (considerando posições de partículas e obstáculos). Uma vez definida a geometria do modelo, realiza-se a leitura paralela do modelo onde cada processo constrói uma porção do domínio correspondente à sua faixa.

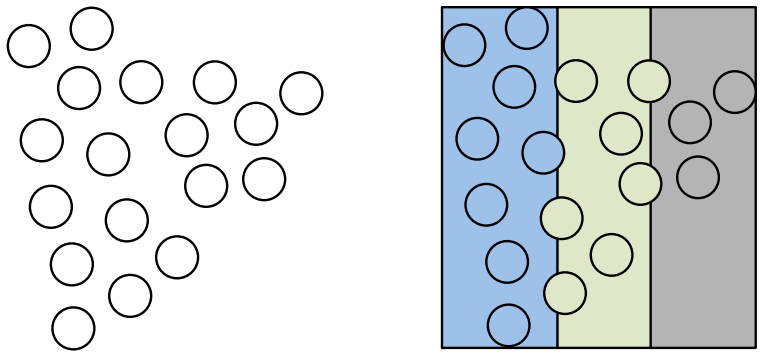
Já vez que a solução do algoritmo do DEM é iterativa (por passos) e envolve cálculos realizados para o conjunto de partículas, prioriza-se então, para cada passo de solução, a sequência:

1. compartilhamento das partículas vizinhas aos subdomínios (comunicação síncrona);
2. definição de partículas do tipo "fantasma"(aquelas contidas no subdomínio vizinho);
3. cálculo de forças de contato envolvendo todas as partículas;
4. determinação das cinemática das partículas do subdomínio;
5. remoção das partículas do tipo "fantasma".

5.4 Paralelização baseada em matrizes de presença

No método de paralelização baseada em matrizes de presença o processo inicia com a geração do modelo. Esta etapa consiste na definição das posições das partículas do modelo, dos parâmetros de interação e demais parâmetros globais de simulação. Os algoritmos de geração de partículas são utilizados nesta etapa para a geração do(s) arquivo(s) de entrada de dados. Assume-se que as dimensões das partículas sejam arbitrárias e definidas na etapa de geração do modelo. A Figura 36 (a) apresenta o domínio inicial do problema apresentado com o intuito de ilustrar a metodologia aqui adotada.

Figura 36 – Geração do modelo.



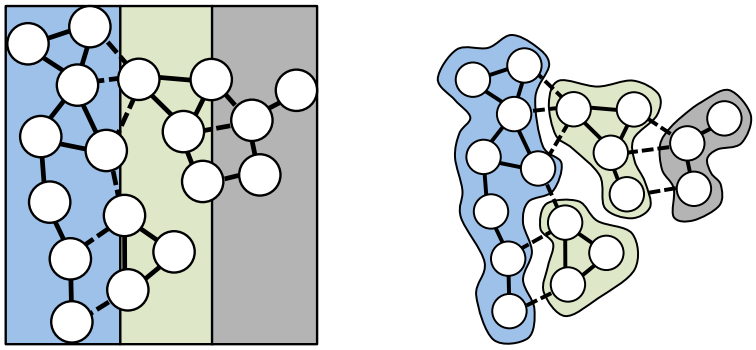
(a) Domínio inicial de partículas. (b) Geração de subdomínios por faixas.

Embora a representação da Figura 36 apresente poucas partículas, em cenários de simulações de larga escala o número de partículas pode ser grande o suficiente para que um único processo não tenha memória suficiente para carregá-lo por inteiro. Faz-se necessário, portanto, que a leitura do modelo seja realizada de forma paralela, conforme ilustra a Figura 36 (b). Cada processo carrega, portanto, uma porção do domínio apresentado. Este procedimento pode ser feito com base em blocos contínuos de partículas ou mesmo conforme a posição geométrica do conjunto. Na Figura 36 (b), a decomposição do modelo inicial é feita através de faixas verticais.

Observe que a distribuição geométrica inicial de partículas não é uniforme, já que o número de partículas é relativamente distinto entre os subdomínios. Para atingir uma melhor performance computacional é necessário que estes subdomínios não apresentem esta característica. Um procedimento de particionamento de domínio mais apurado é então necessário. Como a técnica de particionamento utilizada neste trabalho baseia-se em topologia, a criação dos grafos de interação entre partículas é exigida (Figura 37). As partículas, e suas respectivas posições, constituem os vértices do grafo e as arestas são criadas entre partículas quando suas posições estão abaixo de um determinado valor.

Observe que o grafo gerado, apresentado na Figura 37 (b), é conectado apesar de estar distribuído nos processos. Com os grafos de interação gerados, segue a etapa de particionamento do domínio. Neste trabalho, adota-se o pacote computacional Zoltan para

Figura 37 – Geração do grafo inicial.

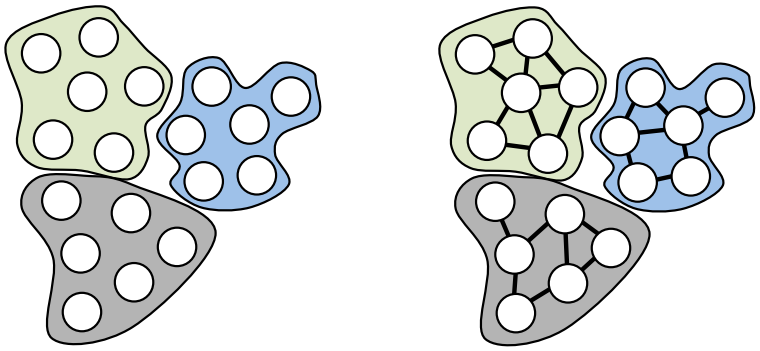


(a) Faixas de referência. (b) Grafo gerado.

o particionamento do domínio (BOMAN et al., 2012). De posse dos dados gerados no particionamento é possível definir os subdomínios correspondentes aos processos, conforme apresenta a Figura 38 (a), onde tem-se um balanceamento de carga e minimização da comunicação.

Neste momento, cada processo gerado visualiza apenas as partículas correspondentes ao seu subdomínio. Pode-se então iniciar o cálculo de forças interativas que surgem dos contatos entre partículas contidas no interior do subdomínio. Este cálculo é feito objetivando a obtenção das forças resultantes atuantes em cada uma das partículas, podendo ser realizado através de uma técnica de busca de contatos qualquer. O grafo de interações para o subdomínio, ilustrado na Figura 38 (b), pode servir de referência para a definição dos pares de contato entre partículas. No final desta etapa, os valores das forças resultantes serão utilizadas na etapa de integração temporal para a definição das novas posições das partículas.

Figura 38 – Particionamento do grafo inicial.

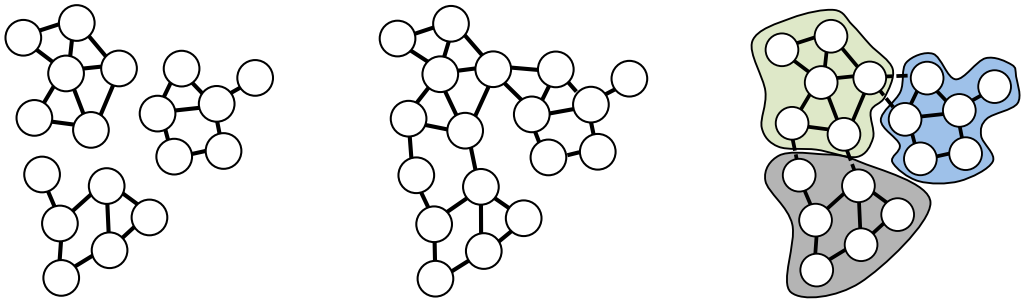


(a) Subdomínios. (b) Sub-grafos.

As partículas que estão situadas na fronteira entre subdomínios, no entanto, precisam ter valores de forças resultantes corrigidos para considerar a interação entre partículas de subdomínios vizinhos. Para tanto é necessário descobrir, dentre os processos, as arestas do grafo global de interações que possuam nós em processos distintos. Isso pode ser feito

tomando como referência o grafo prévio obtido pela decomposição em faixas, conforme ilustra a Figura 39.

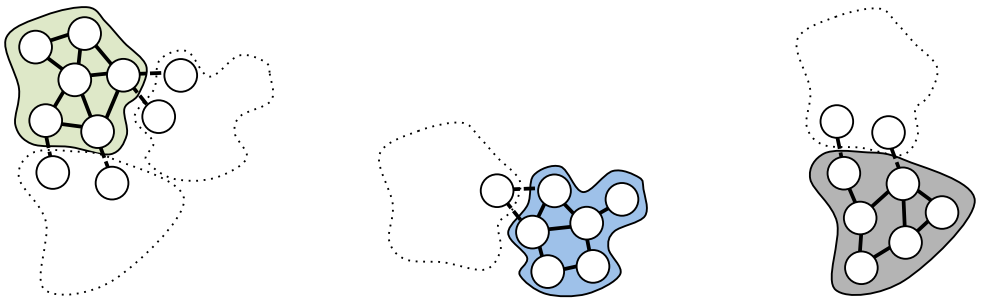
Figura 39 – Geração dos grafos de interação.



(a) Subdomínios. (b) Faixas. (c) Particionado.

Note que, a partir deste momento, os processos responsáveis pelo cálculo dos subdomínios possuem as informações de partículas do seu interior e de suas fronteiras. A Figura 40 apresenta os grafos isolados de cada um dos três subdomínios ilustrativos aqui apresentados. O cálculo da força resultante das partículas das fronteiras pode então ocorrer, precedendo o processo de integração temporal. É interessante que o cálculo das forças de contato do interior do subdomínio seja feito enquanto os processos comunicam-se para compartilhamento das partículas de fronteira. Este procedimento permite processar em paralelo, de forma parcial ou total, a tarefa de comunicação entre processos.

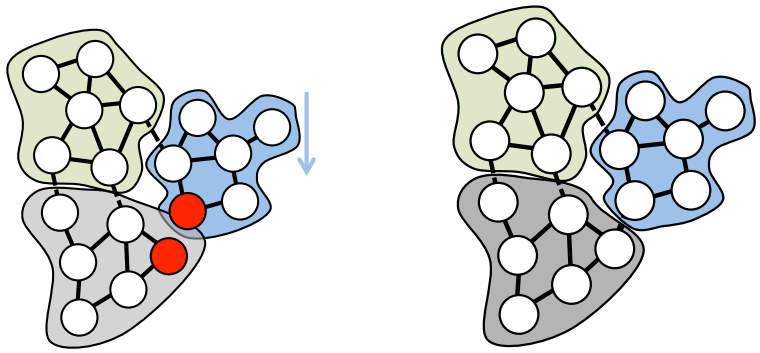
Figura 40 – Grafo de interação dos subdomínios.



(a) Subdomínio 1. (b) Subdomínio 2. (c) Subdomínio 3.

Uma vez realizada a integração temporal, as posições das partículas do modelo mudam e os grafos de interações deixam de ser corretos. Já que a distância é o fator utilizado para a definição das arestas do grafo, quanto menor for seu valor maior é a possibilidade de haver contato entre partículas que não haviam sido mapeadas em uma aresta do grafo. No entanto, como o movimento das partículas é livre, as mesmas podem assumir grandes deslocamentos, logo o grafo de interações precisa ser atualizado. A Figura 41 (a) ilustra a necessidade da criação de uma nova aresta de interação devido a uma simples movimentação relativa entre as partículas. O novo grafo esperado está ilustrado na Figura 41 (b).

Figura 41 – Problema da geração de novas arestas.



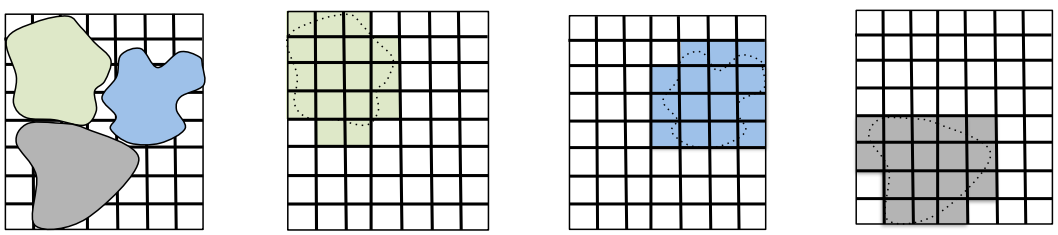
(a) Aresta a ser criada. (b) Grafo desejado.

O processo de atualização do grafo de interações em paralelo não é uma etapa trivial. É preciso lembrar que as informações referentes às partículas estão espalhados entre os processos. No ambiente onde os processos estão inseridos, a comunicação entre eles é realizada através de troca de mensagens, procedimento de alto custo de tempo. No caso específico da Figura 41 (a), a nova aresta surge entre subdomínios sem interação prévia (subdomínios 2 e 3).

Pensando em uma escala maior que a escala das partículas, os subdomínios gerados no processo de particionamento (ou balanceamento de carga) podem ser idealizados como corpos dispersos no espaço com geometria arbitrária. Assim como as partículas, estes corpos movimentam-se e interagem entre si através de forças de contato, de natureza de curto alcance. Uma forma de descobrir a vizinhança dos subdomínios é fazendo seu mapeamento em uma malha estruturada de células.

Inicialmente gera-se, para cada subdomínio, uma malha estruturada de células, em 2 ou 3 dimensões. A malha deve ser construída de forma a conter os limites geométricos de todas as partículas do domínio. As dimensões das células adotadas são arbitrárias, desde que maiores que o diâmetro da maior partícula. Através de um procedimento de mapeamento, armazenam-se referências para as partículas que estão contidas em cada uma das células. Desta estrutura é possível gerar uma matriz de presença de partícula que indica, para cada célula, se existe ou não uma partícula naquela região. O processo de mapeamento de subdomínio em células é ilustrado na Figura 42.

Figura 42 – Mapeamento de subdomínios em células.

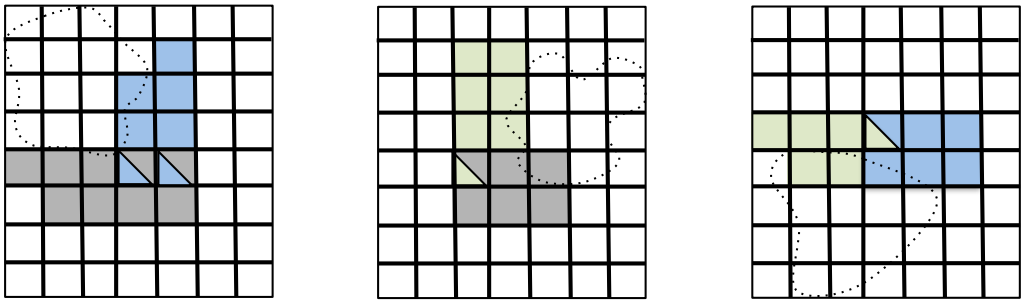


(a) Domínio. (b) Subdomínio 1. (c) Subdomínio 2. (d) Subdomínio 3.

Note que a dimensão da matriz de presença é inversamente proporcional ao tamanho da célula. Isso indica que células grandes contém muitas partículas, no entanto geram matrizes de presença de menor dimensão. O compartilhamento das matrizes de presença entre os subdomínios permite a definição da topologia de comunicação. Para diminuir o custo de comunicação, as matrizes de presença podem ser compostas apenas por elementos booleanos (zero ou um). Seus elementos representam ausência ou presença de partículas nas células correspondentes.

A partir da topologia de comunicação, os processos responsáveis pelo cálculo dos subdomínios podem compartilhar as informações de partículas de contorno entre si. Cada processo tem conhecimento dos processos com quem deve comunicar-se e, inclusive, quais células devem ser solicitadas a cada um deles. Na etapa de solicitação de células, o processo cuja célula foi solicitada envia as partículas mapeadas nas células correspondentes. Os mapas de células solicitadas entre processos para o problema ilustrativo tratado nesta seção é apresentado na Figura 43.

Figura 43 – Células vizinhas nos subdomínios.



(a) Subdomínio 1. (b) Subdomínio 2. (c) Subdomínio 3.

A comunicação para compartilhamento de células é feita somente nas fronteiras entre subdomínios. Uma vez que as células podem assumir dimensões mínimas iguais ao diâmetro da maior partícula do modelo, é necessário o conhecimento apenas das células imediatamente vizinhas para a determinação das forças de contato.

Visto a maneira como os processos comunicam-se, existe uma forte associação entre a dimensão das células e o volume de comunicação demandado nesta metodologia.

Quando as células tem dimensões pequenas, a matriz de presença cresce em tamanho e, consequentemente, impacta negativamente na comunicação. Entretanto, tem-se um detalhamento mais apurado das fronteiras entre subdomínios e a posterior migração de partículas é realizada de forma mais eficiente. Isso ocorre uma vez que os processos apenas enviam para seus vizinhos as partículas de prováveis contatos na fronteira.

Para células maiores, as matrizes de presença são consequentemente menores. Isso contribui positivamente na redução do tempo de comunicação entre processos. No entanto, o volume de partículas posteriormente migrado é maior e com informações em grande

parte desnecessárias.

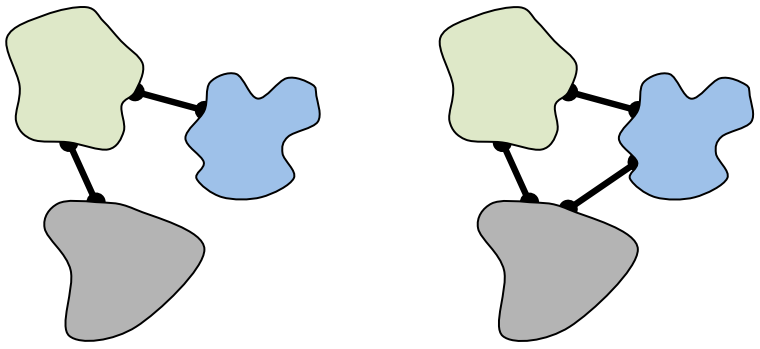
Deve-se, portanto, atentar à escolha das dimensões das células para que haja uma melhor eficiência da estratégia de paralelização empregada.

O processo de atualização de grafo que se baseia em mapeamento de células envolve a comunicação entre os processos e requer a montagem, conforme visto, de uma estrutura de dados auxiliar. Todavia este mecanismo permite a atualização dos grafos de interação necessários no cálculo de forças, bem como no procedimento de balanceamento de carga. Sem a atualização dos grafos, não há como criar arestas entre subdomínios sem interação prévia.

A Figura 44 (a) ilustra a topologia de comunicação criada com base apenas no grafo de interações gerados para os subdomínios. Para esta configuração não é possível detectar o contato entre partículas de alguns dos subdomínios, conforme destacado na Figura 41. Quando a técnica de mapeamento é empregada, no entanto, surge a possibilidade de interação entre subdomínios e a criação de novas arestas. No caso ilustrado nesta seção, a técnica de mapeamento em células gerou uma topologia de comunicação que possibilita a criação de novas arestas, conforme ilustra a Figura 44 (b). As interações entre subdomínios podem ser definidas com base na localização das células ocupadas. Esta informação é obtida dos mapas de presença compartilhados, conforme apresentado na Figura 43.

Dados dois mapas de presença correspondentes a dois subdomínios, definir interação entre eles consiste em buscar contato entre células de subdomínios distintos. Este procedimento é de complexidade linear, com número de operações proporcional ao número de células.

Figura 44 – Topologia de comunicação.



(a) Atualização de grafo baseada em interações prévias.

(b) Atualização de grafo baseada no novo cálculo de distâncias.

Com os grafos de interação atualizados, é possível determinar as forças de interação, realizar a integração temporal das partículas e avançar a simulação ao longo do tempo. Eventualmente, as configurações correntes das partículas podem ser impressas em arquivos para visualização posterior. Admite-se que esta etapa seja realizada de forma paralela para os processos correspondentes aos subdomínios. Cada processo reporta, em seu corres-

pondente arquivo de saída, os dados referentes às partículas contidas no seu subdomínio. Partículas réplicas criadas nas fronteiras dos processos vizinhos não devem ser consideradas. Suas existências são momentâneas e motivadas apenas como artifício de comunicação. O pós-processamento gráfico pode ser realizado com o auxílio de *software* com recurso que permita a visualização de modelos fragmentados em arquivos distintos, a exemplo do ParaView (AHRENS; GEVECI; LAW, 2005b) e DEMView (AMORIM; CINTRA; LIRA, 2006).

O balanceamento dinâmico de carga deve ser realizado acompanhando a tendência de movimento das partículas. Nos estágios mais dinâmicos da simulação, exige-se uma maior frequência de atualização dos particionamentos e de consequente migração das partículas que nos estágio mais estacionários. A seção 3.4 apresenta detalhes do mecanismo de atualização dos particionamentos.

O processo de atualização da topologia de comunicação, cálculo de forças, integração temporal, balanceamento de carga e saída de dados deve ser repetido de forma iterativa até o final da simulação.

5.5 Paralelização combinada (células+topologia)

A técnica de particionamento combinada agrega características das estratégias de divisão de domínio geométrico e topológico. Seu desenvolvimento visa aproveitar boas funcionalidades de ambas as técnicas.

Os algoritmos topológicos possuem vantagens de utilização relacionadas com o balanço de carga e a qualidade de particionamento, medida em volume de comunicação. No entanto, estes algoritmos exigem que a topologia de comunicação seja determinada ao longo dos passos de tempo de integração temporal. Neste trabalho, este procedimento é realizado através de comunicações globais de matrizes de presença de partículas nos processos. Cada processo recebe de todos os demais as suas correspondentes matrizes de presença (comunicação de coleta global). O volume de comunicação pode aumentar consideravelmente dependendo do nível de discretização do modelo numérico considerado.

As técnicas geométricas não requerem que a topologia de comunicação seja definida em todos os passos de integração. Em geral, divide-se o domínio espacial analisado através de células e atribui-se para cada uma delas o processo responsável pela solução das partículas contidas nestes subdomínios. A topologia de comunicação entre processos é definida no instante do particionamento, não alterando até que um novo particionamento seja realizado. Esta atualização comumente é realizada durante a etapa de balanço de cargas, processo que visa equalizar o número de partículas nos processos.

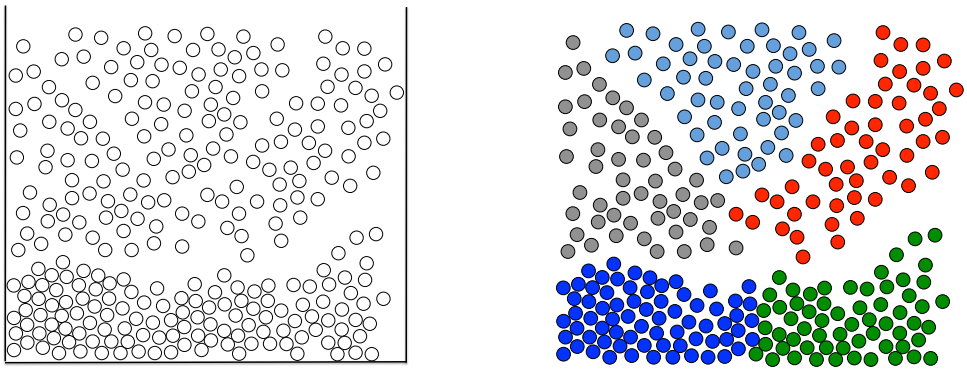
A técnica combinada proposta utiliza dois artifícios empregados nas técnicas de

particionamento geométricas e topológicas. O primeiro artifício é o uso de células de compartilhamento de partículas, utilizado nas técnicas de particionamento geométricos. Neste trabalho, adotam-se células regulares quadradas ou cúbicas. O segundo consiste em empregar técnicas de particionamento topológicos para a divisão do domínio de solução dos processos. A técnica adotada neste trabalho é a de particionamento de grafos em multi-nível. O tamanho das células regulares é arbitrário, desde que maior que três vezes o maior diâmetro dentre as partículas do modelo. Esta condição é necessária para o correto compartilhamento de partículas entre fronteiras de processos. Quando a célula é menor que seu tamanho mínimo, o mecanismo de comunicação de partículas entre fronteiras pode criar múltiplas cópias de um mesmo elemento discreto no mesmo processo.

Em resumo, a técnica combinada define particionamentos tanto para partículas como para as células. No particionamento das partículas, definem-se subconjuntos dos elementos discretos. Considerando m processos, geram-se m subconjuntos de partículas com aproximadamente o mesmo número de elementos. Cada processo é responsável portanto pela solução da dinâmica das partículas de um destes subconjuntos. Como esta solução não pode ser realizada de forma independente, utilizam-se as células espaciais com o intuito de realizar o transporte das partículas para os processos onde suas informações são necessárias.

Para apresentar o mecanismo da técnica combinada, adota-se o cenário hipotético de acomodação de partículas ilustrado na Figura 45 (a). No cenário exposto, o conjunto de partículas é particionado através de uma técnica topológica em subdomínios, representado por cores. O conjunto de partículas, de dispersão espacial qualquer, é então dividido uniformemente entre os subdomínios. Cada partícula recebe então a cor correspondente ao seu subdomínio, conforme Figura 45 (b). Embora neste trabalho adote-se o particionamento topológico de domínio para a definição das cores, a metodologia de particionamento adotada pode ser arbitrária.

Figura 45 – Método de paralelização combinado.

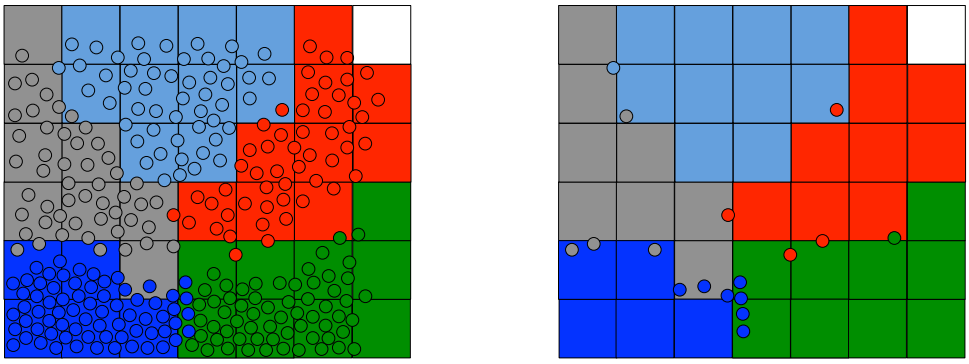


(a) Cenário hipotético de acomodação de partículas.

(b) Particionamento hipotético baseado em topologia para 5 subdomínios (cores).

Para a correta definição da dinâmica das partículas nos subdomínios, é necessário o compartilhamento dos elementos discretos localizados nas suas interfaces. No método combinado, este procedimento é realizado com o auxílio de células geométricas regulares. A cada célula deste conjunto é atribuído um processo responsável pela correspondente região geométrica. Este procedimento é realizado com o intuito de equalizar o volume de partículas que trafegam nos processos. Neste trabalho, propõe-se que o processo escolhido para cada célula seja aquele que possui o maior número de partículas na região delimitada pela célula. A Figura 46 (a) ilustra as células regulares e o procedimento de atribuição de processos.

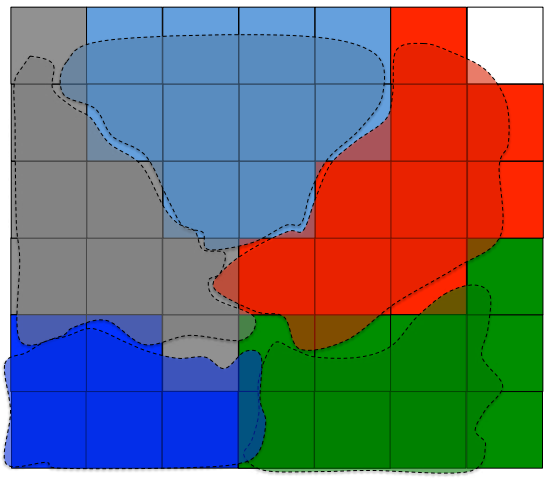
Figura 46 – Células regulares para o método combinado de paralelização.



(a) Atribuição de processos. (b) Partículas intrusas.

Observe que algumas células possuem ou não partículas de apenas um processo. Quando uma partícula está contida em uma célula cujo processo responsável é diferente do seu, esta partícula é denominada como "intrusa". As partículas intrusas obtidas após a atribuição realizada na Figura 46 (a) são destacadas na Figura 46 (b). Os subdomínios irregulares, que definem os limites geométricos do subdomínios, tem a sua geometria aproximada após o procedimento de atribuição de células. A Figura 47 constrasta esta diferença em geometria.

Figura 47 – Aproximação da geometria dos subdomínios por células regulares.



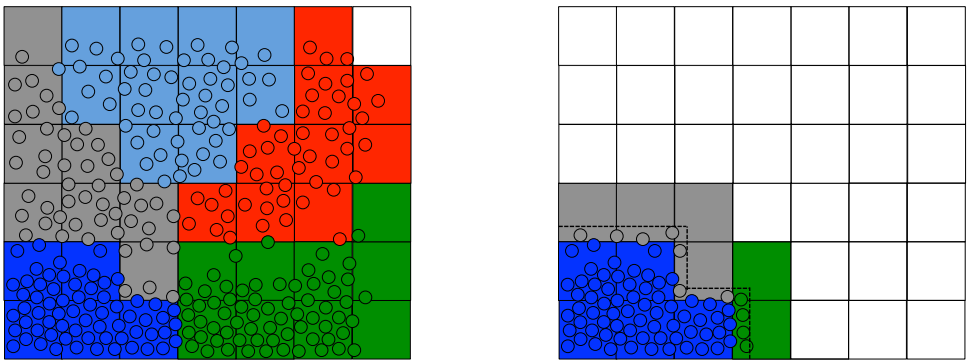
A aproximação em geometria obtida visa facilitar a definição das interfaces entre subdomínios. Este requisito é essencial no procedimento de comunicação entre processos. As interfaces são necessárias para definir quais partículas devem ser enviadas para os processos visando o cálculo de interações do DEM.

A estratégia de divisão em células deve considerar o domínio delimitante do modelo de partículas. Como as partículas movimentam-se ao longo do tempo de simulação, sugere-se que o domínio delimitante englobe todo o provável espaço ocupado pelas partículas durante a simulação. A utilização das células regulares para a de comunicação de partículas segue metodologias distintas apresentadas nas seções 5.5.1 e 5.5.2.

5.5.1 Mapeamento de particionamento em células

Na estratégia de comunicação baseada em mapeamento de particionamento, as células são utilizadas com o intuito de incorporar partículas intrusas. Este mecanismo permite que o cálculo da dinâmica das partículas que estão contidas nas células seja realizado por apenas um processo. Este procedimento perturba dinamicamente o particionamento do domínio. Isto ocorre uma vez que as partículas migram de um processo para outro quando se deslocam entre células de processos diferentes. A Figura 48 (a) ilustra o desbalanço de particionamento obtido em virtude da incorporação de partículas. Note que a mesma é ligeiramente diferente da Figura 46 (a) em decorrência da inclusão das partículas intrusas.

Figura 48 – Mapeamento de particionamento em células.



(a) Desbalanço de particionamento (inclusão de partículas intrusas).

(b) Compartilhamento de partículas de interface.

Para este método de comunicação, o compartilhamento de partículas entre subdomínios é feito apenas através das interfaces geométricas entre células. A Figura 48 (b) retrata este procedimento de compartilhamento de partículas por vizinhança. Um dos processos é adotado como referência neste caso e as células vizinhas pertencentes a processos distintos são selecionadas. Dentre as células vizinhas, coletam-se as partículas situadas nas localidades próximas à interface. Estas partículas devem estar localizadas a uma distância

mínima igual ao diâmetro da maior partícula do modelo. O procedimento de coleta de partículas vizinhas repete-se, de forma paralela, para os demais processos.

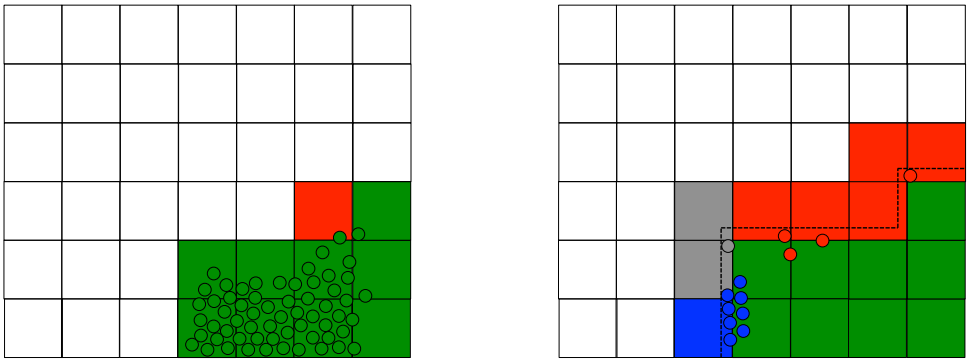
A vantagem desta estratégia de comunicação é que as partículas de fronteira entre processos são compartilhadas em apenas uma etapa. Entretanto, a qualidade de particionamento pode ser afetada em virtude da incorporação das partículas intrusas.

5.5.2 Redirecionamento baseado em células

No método de comunicação baseado em redirecionamento, as células são utilizadas com o intuito de definir os processos responsáveis por redirecionamentos de partículas. Este mecanismo permite que o particionamento não seja alterado em virtude do movimento das partículas, como ocorre na metodologia apresentada na seção 5.5.1. Para tanto é necessária uma tarefa auxiliar de busca de vizinhança de partículas.

Inicialmente, cada processo possui o conjunto de partículas definido pelo procedimento de particionamento de domínio, conforme ilustra a Figura 49 (a). Para cada instante de integração, a comunicação das partículas de fronteira entre processos ocorre em duas etapas. Na primeira etapa, transportam-se as partículas para os processos correspondentes às células espaciais, considerando uma tolerância para vizinhança imediata (Figura 49 (b)).

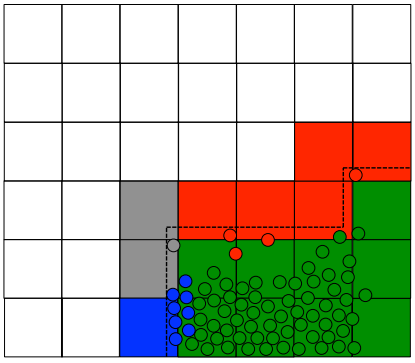
Figura 49 – Redirecionamento baseado em células - Etapa 1.



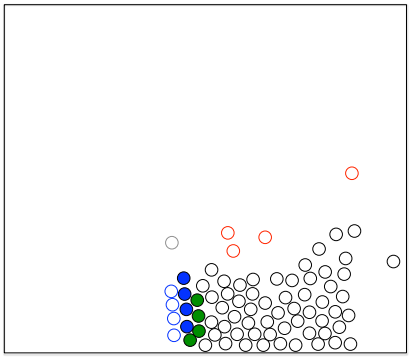
(a) Configuração após particionamento. (b) Transporte de partículas na etapa 1.

Na segunda etapa, realiza-se uma busca de vizinhança para definir com precisão para quais processos as partículas devem ser enviadas de fato. Esta busca visa identificar pares de partículas provenientes de processos distintos. Nesta estratégia de comunicação, as partículas de fronteira são recebidas nos processos que demandam suas informações durante a primeira ou segunda etapa de envio. A Figura 50 ilustra a etapa de redirecionamento que utiliza o artifício de busca de vizinhança para comunicação de partículas.

Figura 50 – Redirecionamento baseado em células - Etapa 2.



(a) Configuração após etapa 1.



(b) Busca de vizinhança para redirecionamento de partículas.

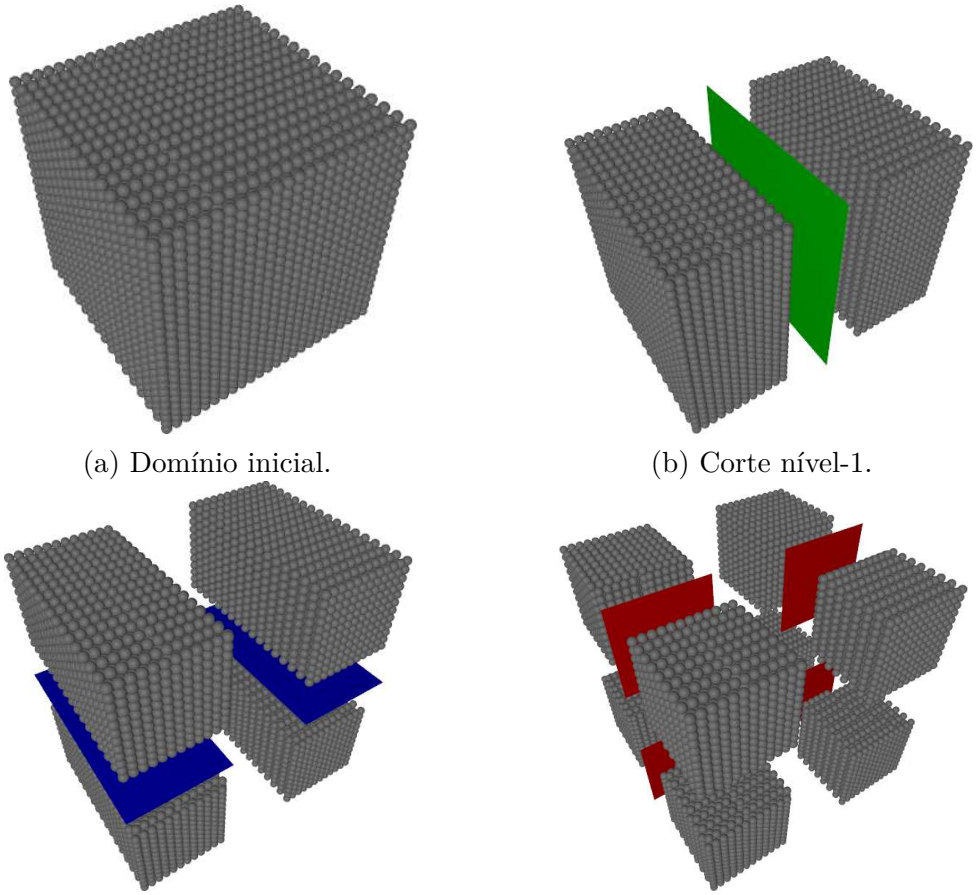
A vantagem desta metodologia é o melhor controle do particionamento das partículas entre os subdomínios. Entretanto, a comunicação em duas etapas pode oferecer restrições em virtude das trocas de mensagens adicionais entre processos.

5.6 Paralelização baseada em RCB

Em simulações do DEM, o custo computacional mais significativo deriva da determinação do movimento das partículas. Uma vez que as interações entre estes elementos ocorrem devido ao contato, os métodos de particionamento tendem a criar subdomínios aglomerados de partículas. Esta configuração reduz as interações entre partículas de processos distintos uma vez que elas acontecem somente através das fronteiras, ao invés de todo o domínio interior.

O Método de Bisseção Recursiva de Coordenadas (RCB - *Recursive Coordinate Bisection*) é um método de particionamento que utiliza as posições de um conjunto de objetos para criar subdomínios espaciais, conforme apresentado na subseção 3.1.2. A técnica particiona o domínio através de planos de corte sempre ortogonais aos eixos coordenados. Embora a qualidade das partições seja mais pobre que as obtidas pelos métodos topológicos, seu uso em aplicações do DEM é bastante apropriado. Isso ocorre uma vez que a conectividade entre elementos muda frequentemente durante a simulação neste método sem malha. O emprego de técnicas baseadas em topologia requer, nestes casos, uma constante atualização de particionamentos. Isso pode acarretar em perda de eficiência do sistema paralelo. As partições baseadas no método RCB, em geral, tendem a ser incrementais e portanto não exigem a movimentação de um volume expressivo de partículas durante o processo de particionamento. A Figura 51 ilustra a sequência de cortes do método RCB para um conjunto arbitrários de objetos, apresentados como uma coleção de partículas.

Figura 51 – Particionamento de domínio usando o método RCB.



(a) Domínio inicial. (b) Corte nível-1.

(c) Corte nível-2. (d) Corte nível-3.

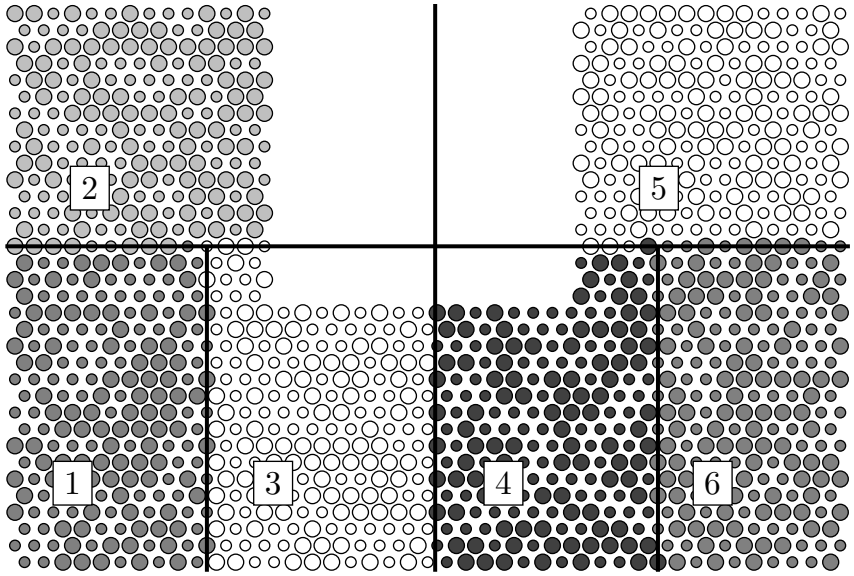
O método RCB é simples de implementar e executa rapidamente. Uma implementação que executa em sistemas paralelos com memória distribuída está disponível na biblioteca Zoltan (BOMAN et al., 2012). As implementações realizadas nestes trabalho, que utilizam particionamentos baseados no método RCB, utilizam a biblioteca Zoltan para particionar o domínio.

5.6.1 Estratégia de comunicação de partículas

Em um sistema de memória distribuída, todas as informações a respeito das partículas que compõem o domínio estão armazenadas na memória do processo que as contém. Cada partícula pertence a um único processo responsável pela atualização de suas posições ao longo do tempo. No entanto, uma vez que as partículas são livres para movimentar-se no espaço, o número de partículas em cada processo deve alterar durante a simulação uma vez que as fronteiras entre subdomínios permanecem constantes após o procedimento de particionamento. Estas mudanças no número de partículas ocorrem uma vez que estes elementos movimentam-se para regiões do espaço físico que foram atribuídas para processos diferentes.

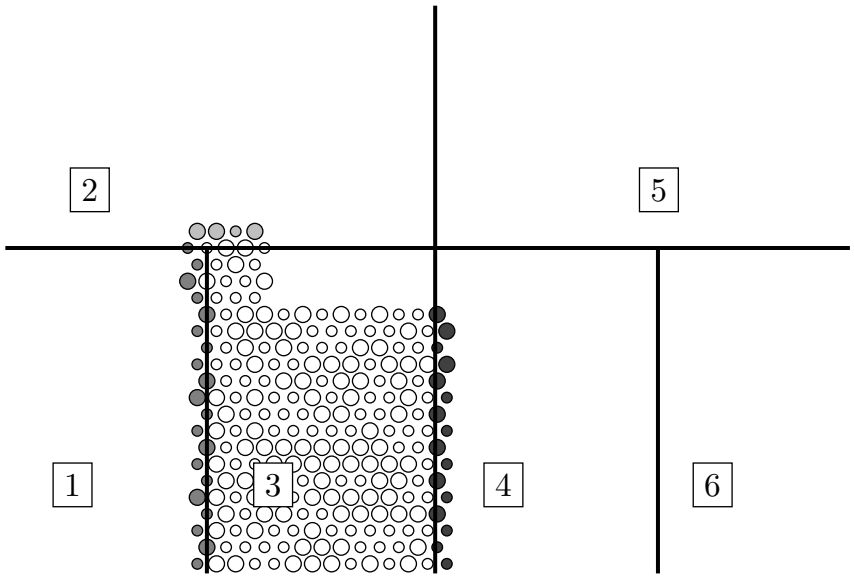
O movimento no espaço físico deve ser acompanhado por trocas de partículas entre os correspondentes subdomínios computacionais, sendo implementadas através de mecanismos de migração. Quando se utiliza o método RCB, uma partícula migra entre processos quando a mesma sai do domínio espacial do processos original. A partícula é então removida do processo de origem e criada no processo que a recebe. Este tipo de comunicação é necessário para que a estratégia de particionamento funcione devidamente e para que o cálculo paralelo de forças do DEM seja correto. Entretanto, uma vez que o número de partículas varia no tempo para cada processo, pode-se gear um desbalanço de carga na simulação paralela do domínio. A Figura 52 ilustra um domínio hipotético dividido em seis subdomínios e suas correspondentes partículas.

Figura 52 – Particionamento de domínio para seis subdomínios bidimensionais.



Outro tipo de comunicação envolvida na simulação paralela através do método RCB é o de interação entre subdomínios vizinhos. Esta comunicação visa o cálculo das forças de interação entre partículas que estão em contato com outros elementos discretos localizados em processos vizinhos. Esta tarefa é realizada para as partículas situadas próximas das interfaces geométricas entre subdomínios. O procedimento de comunicação é necessário uma vez que o movimento de cada partícula é regido pela interação com todas as partículas que a afetam, mesmo quando estas estão localizadas em subdomínios computacionais diferentes. Trata-se basicamente de um procedimento de cópia temporária. Cada processo copia para a sua memória as informações das partículas dos processos vizinhos. Estas cópias de partículas podem ser inclusive armazenadas no vetor de partículas do domínio, entretanto sinalizadas com algum atributo que indica que a mesma deve ser removida da memória após o procedimento de cálculo de forças interativas. A Figura 53 destaca em tons de cinza as partículas vizinhas do subdomínio (3), apresentado na Figura 52. Estas partículas são provenientes dos subdomínios 1, 2 e 4.

Figura 53 – Comunicação de partículas vizinhas usando o método RCB.



Ambas as comunicações de migração de partículas e de compartilhamento de vizinhos são implementadas através de trocas de mensagens entre processos usando o padrão MPI (SNIR et al., 1998). Em um dado passo de tempo de simulação, cada processo envia mensagens para os demais indicando os volumes de comunicação. Trata-se de uma comunicação global que contém o número de partículas envolvidas (aquelas que migram e as vizinhas) e informações relacionadas, tais como as quantidades de contatos anteriores de cada partícula (necessários para o cálculo de forças tangenciais). Depois deste estágio de comunicação, cada processo aloca os recursos de memória necessários para receber as partículas e prepara as comunicações ponto a ponto entre os processos. As partículas são então comunicadas e, a partir deste instante, cada processo possui partículas que já os pertencia e novas partículas. Realiza-se então a classificação das partículas a fim de distinguir quais delas serão removidas após o cálculo de forças e quais permanecem. Uma forma simples de fazer a classificação das partículas é usando um atributo que indica se a partícula pertence ao interior do domínio ou não. As partículas vizinhas podem ser identificadas após o cálculo de forças através deste atributo.

6 Estratégias de paralelização híbrida (MPI+OpenMP)

Os ambientes computacionais de memória compartilhada, em geral, lidam com recursos de mais fáceis implementações paralelas, dentre os quais pode-se citar o OpenMP. No entanto, estas arquiteturas limitam a escala do problema devido a questões de memória. A memória disponibilizada por estes ambientes pode não ser bastante para armazenar todo o modelo e suas estruturas de dados em problemas de larga escala. Os sistemas distribuídos superam esta questão através do uso de protocolos de troca de mensagens, tais com o MPI. Eles também permitem o desenvolvimento de sistemas híbridos que combinam ambientes de memória distribuída e compartilhada.

Os sistemas híbridos têm demonstrado alta performance em muitas simulações de partículas para sistemas heterogêneos incluindo CPUs (SHIGETO; SAKAI, 2011; KUNASETH et al., 2013) e GPUs (CHEN et al., 2009). Experimentos envolvendo sistemas híbridos relatam ganhos significativos em relação às execuções baseadas unicamente em MPI. Isto ocorre devido ao melhor controle de balanço de carga e aos menores volumes de comunicação envolvidos. De qualquer maneira, um problema comum no desenvolvimento destes sistemas é evitar as condições de corrida que acontecem devido ao cálculo de forças de contato realizado através de *threads*.

Trabalhos anteriores apresentaram modelos híbridos de programação paralela para o algoritmo do DEM. A maioria das estratégias consideram a terceira lei de Newton visando reduzir o número total de testes de intersecção e cálculos de força. Estes métodos precisam controlar os acessos a memória realizados pelas *threads* visando banir operações simultâneas impróprias. Os mecanismos empregados incluem a replicação do vetor de forças por *threads*, utilização de tabelas contendo pares de contatos (NISHIURA; SAKAGUCHI, 2011) e a utilização de diretivas de compilação que asseguram operações atômicas (HENTY, 2000),ou seja, enquanto um processo estiver executando uma operação, nenhum outro processo pode executar outra operação. Estes procedimentos adicionam *overheads*[1] ao tempo de processamento computacional devido a operações de redução para o vetor de forças ou devido às travas (semáforos) utilizadas na sincronização das *threads*.

Berger et al. (2015) apresentam uma estratégia híbrida de cálculo paralelo do DEM que não utiliza travas e nem operações de redução. O método utiliza uma técnica de particionamento local baseada na decomposição RCB visando a divisão do domínio

[1] *overhead* - Qualquer processamento ou armazenamento em excesso, seja de tempo de computação, de memória, de largura de banda ou qualquer outro recurso que seja requerido para ser utilizado ou gasto para executar uma determinada tarefa.

computacional entre subdomínios e também em subconjuntos de partículas. A divisão em subdomínios é realizada no nível de memória distribuída enquanto que os subconjuntos de partículas são definidos para cada subdomínio e atribuídos para cálculo através de *threads*. O método adotado realiza duplicação de trabalho apenas nas interfaces entre subconjuntos de partículas, ao invés de todo o interior do subdomínio. O método atinge boa eficiência paralela mas adiciona um *overhead* considerável ao processamento devido a operações de memória necessárias para a validação das partições em tempo de execução.

Nesta tese, uma estratégia híbrida de paralelização que considera as curvas de preenchimento de espaço de Hilbert é adotado para o particionamento local dos subdomínios. A metodologia proposta permite combinar o particionamento local de partículas com métodos de otimização de acesso à memória e de balanço de carga. Uma vez que o procedimento de particionamento utiliza um vetor de partículas com localidade espacial melhorada, as interfaces entre subconjuntos de partículas são reduzidas. Este processo também elimina a etapa de validação de partições durante os passos de tempo de solução, eliminando os *overheads* correspondentes.

6.1 Aspectos gerais

Em geral, execuções paralelas de simulações do DEM exigem um vasto conjunto de operações para calcular a cinemática simples de partículas individuais. Esta exigência é principalmente motivada para natureza discreta do procedimento numérico. No DEM, existe uma inconstância de conectividade entre elementos uma vez que os contatos são criados e destruídos para cada partícula no domínio durante a simulação.

A definição dos contatos é essencial para o cálculo de forças de interação no método. Este procedimento também direciona a maioria dos cálculos no DEM e pode ser realizado de forma paralela. No modelo *soft-sphere*, o contato é definido por superposições espaciais entre objetos: partículas ou obstáculos. A busca de contatos visa encontrar pares de objetos potencialmente próximos. Os pares encontrados na busca de contato devem passar por testes geométricos de superposição antes do cálculo de forças ser realizado. Um algoritmo de busca de contato eficiente descarta testes geométricos desnecessários. Isto aumenta a performance computacional especialmente para modelos com um número grande de partículas.

Os métodos de decomposição de domínio fornecem recursos bastante úteis para a simulação paralela de partículas. Usualmente estes métodos operam dividindo o conjunto de partículas, que basicamente representa o esforço de cálculo do DEM. Cada subdomínio gerado é atribuído para um processador que é responsável pelos cálculos relacionados com a cinemática das partículas. O número de subdomínios gerado pode ser arbitrário, mas em geral o número de processadores disponível é escolhido no intuito de reduzir volumes

de comunicação entre processos. Os métodos de decomposição empregados comumente utilizam estratégias de particionamento que agrupam partículas em subdomínios por proximidade. Estas estratégias podem considerar, como visto anteriormente, informações tanto geométricas como topológicas.

Embora o objetivo da decomposição de domínio seja criar subdomínios disjuntos de cálculo independente, na maioria dos casos isto não é possível. Portanto, a decomposição precisa criar um conjunto de partículas internas e dinamicamente descobrir e atualizar um conjunto auxiliar de partículas vizinhas para cada subdomínio.

O cálculo paralelo de cada subdomínio demanda então a descoberta e a comunicação das partículas vizinhas. Cada processo responsável pelo cálculo do subdomínio executa as atividades mencionadas como uma exigência para o cálculo coletivo de solução do domínio. As partículas vizinhas, também chamadas de partículas fantasmas, introduzem um *overhead* de comunicação entre processos (subdomínios), operações de memória no vetor de partículas e, portanto, processamento adicional.

Este modelo de paralelização descreve em termos gerais a metodologia de solução paralela baseada em troca de mensagens. Pode-se observar que o *overhead* causado pela existência de partículas vizinhas entre processos aumenta com o número de subdomínios. Em uma situação hipotética em que o número de subdomínios é igual ao número de partículas, o número de partículas vizinhas em cada subdomínio corresponde ao número de contatos de cada partícula. Neste cenário, as tarefas de comunicação são dominantes e o tempo de processamento é altamente afetado por estes procedimentos de cara execução.

6.2 Paralelização híbrida

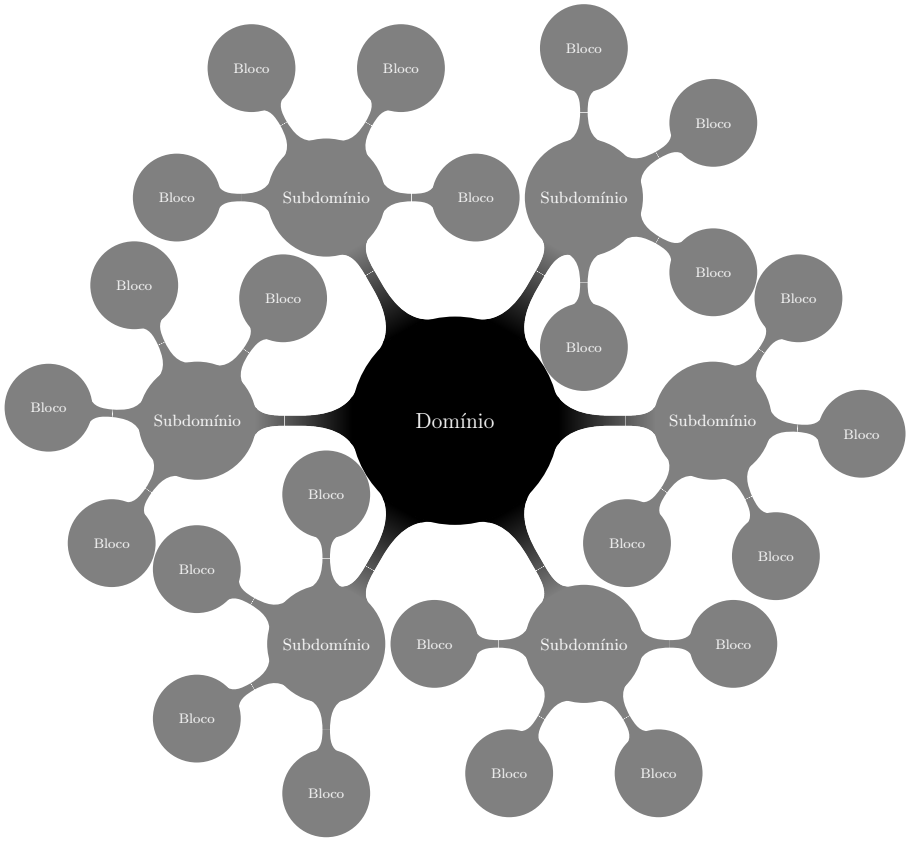
A ideia principal do modelo de paralelização híbrido é combinar o uso de modelos distintos de paralelização. Esta metodologia é especialmente útil quando os recursos de *software* precisam ser empregados em ambientes de *hardware* heterogêneos. No passado, os *clusters* de computadores eram compostos por um conjunto de nós contendo processadores de um único núcleo. Esta arquitetura de *hardware* guiou durante anos o desenvolvimento de *software* baseado em troca de mensagens. O MPI (*Message Passing Interface*) é ainda vastamente utilizado em diversas rotinas computacionais (SNIR et al., 1998).

Atualmente, os *clusters* contém nós de cálculo dotados de CPUs de múltiplos núcleos, GPUs e aceleradores. O OpenMP então surgiu como um modelo portável e escalável para paralelização *multi-thread* (DAGUM; MENON, 1998). Sua API pode ser aplicada em diversas plataformas, desde computadores pessoais a supercomputadores.

Devido à sua portabilidade, o OpenMP pode ser utilizado em conjunto com o MPI em aplicações construídas com o modelo híbrido de programação paralela. No contexto

de simulação de partículas, o modelo híbrido pode ser idealizado tal como a Figura 54 apresenta.

Figura 54 – Modelo de paralelização híbrido.



Neste modelo, o domínio global de solução que contém o conjunto completo de partículas pode ser decomposto através de um método de particionamento em subdomínios distintos para execução paralela. A solução de cada subdomínio pode ainda ser realizada utilizando *multi-threading*. Uma vez que o procedimento de busca de contatos constitui a porção principal do tempo de execução no DEM, é natural a utilização de *threads* na realização destas tarefas. Para tanto o conjunto de partículas do subdomínio é dividido em blocos.

Com o objetivo de evitar condições de corrida (concorrência de acesso à memória), situações em que *threads* diferentes tentam escrever ou ler a memória desordenadamente, é necessário guiar o acesso à memória através de um método de particionamento local. Esta tarefa de agrupamento atribui um rótulo (cor) para cada partícula e ordena o vetor de partículas por esta informação. Depois da ordenação, cada *thread* busca contatos para as partículas de cada rótulo. Neste trabalho, diferentes métodos de particionamento local foram escolhidos com o intuito de comparação de performance. Estes métodos são discutidos em mais detalhes na seção 6.4.

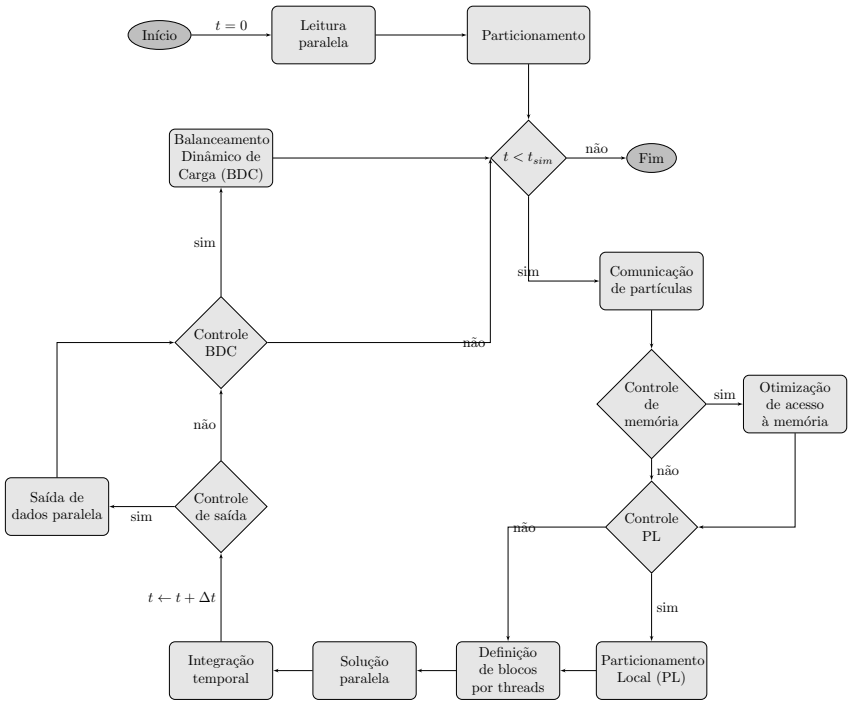
O modelo híbrido de paralelização funciona bem quando a porção estritamente paralela do programa é alta. Caso contrário, os ganhos de tempo são limitados tal

qual sugere a lei de Amdahl (MCCOOL; ROBISON; REINDERS, 2012). Desta forma, para o sucesso do emprego do paralelismo é essencial otimizar tarefas adicionais de particionamento, ordenação e procedimentos seriais do código.

6.3 Workflow

Existem diversas formas de implementar o modelo de paralelização híbrida usando o DEM. Cada método pode combinar o uso de um conjunto vasto de algoritmos com propósitos distintos. Do ponto de vista geral, estes algoritmos são desenvolvidos assumindo um fluxo de trabalho que compõe um método de marcha temporal ao longo do tempo de simulação do DEM. A Figura 55 apresenta o fluxo de trabalho utilizado nas soluções de paralelização híbrida adotadas neste trabalho.

Figura 55 – Estratégia paralela híbrida.



De forma semelhante ao fluxo de trabalho apresentado na Figura 34, a simulação inicia com a leitura paralela do domínio e o particionamento imediato de domínio. Esta tarefa define todos os subdomínios de solução e distribui-os para os processos. As técnicas de particionamento de domínio foram discutidas previamente no Capítulo 3, não sendo tratadas novamente neste capítulo do trabalho. No entanto, cabe salientar que esta atividade permite que um problema de larga escala possa ser dividido em problemas menores de solução paralela.

Depois da operação de particionamento, a rotina entra no processo de repetição que ocorre durante a simulação. A solução do DEM avança ao longo do tempo através de

passos de tempo no método de marcha temporal regido pelo processo de integração.

Durante cada passo de tempo, é necessário o intercâmbio de partículas entre processos visando o cálculo correto das forças de interação entre partículas de processos diferentes. Este procedimento envolve as troca de mensagens entre processos porque as informações de partículas vizinhas estão armazenadas em memórias distribuídas sem acesso direto através de *threads*.

Uma vez que cada subdomínio conhece suas partículas e vizinhos, a solução do DEM não exige mais comunicação entre processos neste estágio e segue sua solução sequencial. Procedimentos auxiliares de controle de memória podem então ser realizados neste momento conforme discutido na subseção 4.3.2.

A metodologia descrita até então em nada difere do modelo de paralelização baseado unicamente em trocas de mensagens. Note que o *workflow* apresentado na Figura 55 é bastante semelhante ao mostrado na Figura 34. As diferenças observadas ocorrem devido às atividades de particionamento local e definição de *threads*, discutidas nas seções 6.4 e 6.5 respectivamente. Estas atividades permitem a utilização do recurso de *multi-threading*.

6.4 Particionamento local

O procedimento de particionamento local é uma técnica que define blocos de trabalho para serem processados por *threads*. O método é aplicado no cálculo paralelo do DEM, para os modelos de paralelização baseados em OpenMP ou em um modelo híbrido, uma vez que ele cria partições para o vetor de partículas.

Dependendo de como a rotina computacional considera o evento de contato, este particionamento pode ser uma tarefa trivial ou demandar o uso de um método de agrupamento na decomposição. O caso trivial acontece quando a busca de contatos induz o cálculo de cada partícula de forma individual. Então, se dois objetos A e B entram em contato durante a simulação, tanto o teste de intersecção geométrica quanto o cálculo de forças acontecem para os objetos A e B. Isto gera uma duplicação de trabalho que pode ser evitada assumindo a terceira lei de Newton, que define ação e reação. Deve-se ter em mente que a carga de computação cai pela metade quando assumimos que o valor do força de contato apenas precisa ser calculada uma vez para cada par.

Outra metodologia aplicável consiste em agrupar partículas por partição e atribuir o cálculo dos grupos às *threads*. Os contatos que acontecem entre objetos do mesmo grupo aplicam ação e reação nos objetos A e B. A força deve ser calculada apenas uma vez para o par e aplicada com sinais opostos nos objetos. Uma forma simples de evitar a duplicação de trabalho é através de identificadores sequencias atribuídos aos objetos. Para tanto, cada partícula armazena em sua estrutura de dado um valor numérico de identificação de tal

maneira que duas partículas distintas não tenham o mesmo número. A própria posição da partícula no vetor de partículas pode ser utilizada com este objetivo. Quando o algoritmo de busca gera um par de partículas do mesmo grupo, o teste de contatos e o cálculo de forças apenas devem ser realizados caso o identificador do objeto A seja menor que o do objeto B (ou vice-versa). Em caso contrário, nada deve ser feito para o par gerado.

Quando o algoritmo de busca gera um par de partículas que pertencem a grupos distintos, as forças de interação são calculadas individualmente e aplicadas em apenas um dos objetos. Desta forma, uma *thread* atualiza a força no objeto A e outra *thread* atualiza no objeto B. Existe portanto duplicação de trabalho apenas nestes casos de interface entre grupos. A eficiência deste método de cálculo é melhor quando as interfaces entre partições são menores. Isto geralmente pode ser obtido através de métodos de particionamento que consideram agrupamento por proximidade.

A Figura 56 ilustra o procedimento de particionamento local. Na Figura 56 (a), as partículas do subdomínio 3, obtido da decomposição de domínio apresentada na Figura 52, são isoladas. Depois do particionamento local, estas partículas recebem as correspondentes etiquetas que definem suas partições. A Figura 56 (b) apresenta uma ilustração dos grupos obtidos e destacados por suas etiquetas.

Figura 56 – Particionamento local.

(a) Partículas do subdomínio. (b) Grupos obtidos.

As mesmas técnicas de particionamento discutidas no Capítulo 3 podem ser utilizadas para o particionamento local de subdomínio. Na seção 6.6 serão discutidas algumas estratégias de paralelização híbrida baseadas em métodos distintos de particionamento local.

6.5 Definição de blocos por *threads*

As rotinas de particionamento local geram uma etiqueta para cada partícula no subdomínio. Entretanto, esta atividade não necessariamente ordena o vetor de objetos com base nos dados gerados. A ordenação do vetor de objetos é necessária porque os

procedimentos de busca de contato precisam conhecer os blocos de partículas que cada *thread* deve processar. Estes blocos informam as posições inicial e final do vetor de partículas.

A definição dos blocos acontece para cada passo de tempo de solução uma vez que o vetor de partículas tem características dinâmicas. Este comportamento dinâmico é decorrente do processo migratório de partículas entre processos. Dentro de cada bloco de partículas, apenas devem existir objetos com a mesma etiqueta. Isto torna a partição (bloco) válida e permite o cálculo paralelo através de *threads* sem conflitos de acesso à memória.

Operações de permutação ou ordenação podem ser realizadas para o vetor de partículas visando a definição dos blocos dependendo de como o particionamento local ocorra. Estas operações podem ter custo computacional relevante e adicionar *overheads* ao processamento. Berger et al. (2015) apontam a operação de validação de particionamento como sendo um ponto fraco do método híbrido desenvolvido em seu trabalho. Uma solução para este problema é apresentada como contribuição desta tese na subseção 6.6.3.

6.6 Estratégias de solução

A partir da metodologia geral apresentada na Figura 55, é possível especificar diferentes estratégias de solução paralela híbridas combinando algoritmos distintos de decomposição de domínio, controle de memória e particionamento local. Nesta seção, os métodos de solução adotados nesta tese são apresentados. Visando conduzir os estudos comparativos de eficiência paralela realizados neste trabalho, três estratégias de solução foram adotadas com base nos métodos de particionamento local escolhidos. Para todas as estratégias de solução, o particionamento de domínio baseado no método RCB foi utilizado na criação dos subdomínios de solução. As estratégias de controle baseadas em deslocamento limite e desbalanço de particionamento foram adotadas com o objetivo de balanço dinâmico de carga. Todas as implementações computacionais foram realizadas no *software* DEMOOP.

O primeiro método de solução, descrito na subseção 6.6.1, refere-se à estratégia de paralelização híbrida apresentada por Berger et al. (2015). Este método também utiliza o particionamento RCB para a definição das partições locais de cada subdomínio. A segunda metodologia adotada é baseada em particionamento topológico e fundamenta-se na definição de grafos criados com base na vizinhança das partículas (subseção 6.6.2). Por fim, um método combinado de controle de memória, balanço de carga e particionamento é então apresentado na subseção 6.6.3.

6.6.1 Particionamento local por RCB

O método RCB de particionamento local é realizado utilizando o método de decomposição geométrica de mesmo nome. Assim, tanto a tarefa de decomposição de domínio quanto a de particionamento local utilizam a sequência de divisão recursiva visando a definição de subdomínios e grupos de partículas. Esta implementação do modelo híbrido de programação foi apresentada anteriormente por Berger et al. (2015). Ela considera o método de controle de memória PSS (subseção 4.3.1) visando a criação dos grupos de partículas atribuídos para processamento através de *threads* e a melhoria de uso da memória *cache*. Este método baseia-se em tarefas de particionamento, alteração de partições existentes e permutações de memória.

A tarefa de particionamento, referente ao método RCB propriamente dito, atribui para cada partícula uma etiqueta que identifica a partição da mesma. Neste contexto, uma partição pode ser entendida como uma caixa limite que delimita uma porção geométrica do subdomínio. Uma vez que as partículas alteram suas posições durante a simulação, elas podem mover-se de uma partição para outra gerando perturbações no balanço de carga de computação das *threads*. Esta migração aumenta ainda o número de testes geométricos nas interfaces entre *threads* e devem ser evitadas. Um reparticionamento local completo deve ser disparado visando balancear a carga de trabalho das *threads*. Enquanto isto não é realizado, as partículas são atribuídas às partições existentes. Berger et al. (2015) não especificam a frequência de reparticionamento local adotadas em seus experimentos numéricos. Neste trabalho o critério de deslocamento limite foi adotado para definir a frequência de atualização das partições locais. Para tanto, considera-se o deslocamento limite igual a $2r_{\max}$.

Tanto o reparticionamento quando o movimento de partículas alteram o vetor de partículas através desta metodologia. Estes eventos exigem operações de memória com o objetivo de validar os blocos de partículas atribuídos às *threads*. Neste trabalho, adota-se a ordenação do vetor de partículas por identificador de partição com o objetivo de validar estes blocos. A contínua ordenação das informações de partículas é custosa computacionalmente para este método de particionamento local, sendo apontada por Berger et al. (2015) como um ponto negativo desta técnica.

6.6.2 Particionamento local topológico (METIS)

Enquanto o método RCB utiliza decomposição geométrica para definir as partições locais, o método METIS considera um método topológico de partição. Esta estratégia cria e particiona um grafo que define uma topologia. A definição do grafo aqui adotada é baseada na vizinhança das partículas. Desta forma, cada partícula no subdomínio representa um nó do grafo. As arestas do grafo conectam duas partículas, existindo quando a distância entre centros é menor que $2,5r_{\max}$, sendo $r_{\max}$ o raio máximo das partículas do subdomínio. O

método METIS utiliza a biblioteca METIS (KARYPIS; KUMAR, 1998a) para particionar o grafo e definir as partições locais de partículas. Esta biblioteca é amplamente utilizada na comunidade científica e produz partições de alta qualidade.

O método de solução baseado em METIS também baseia-se na ordenação contínua dos dados de partículas. No entanto, a metodologia de solução baseada em particionamento de grafo é diferente do método RCB. Isto deve-se ao fato de que não existe migração de partículas entre partições neste método topológico.

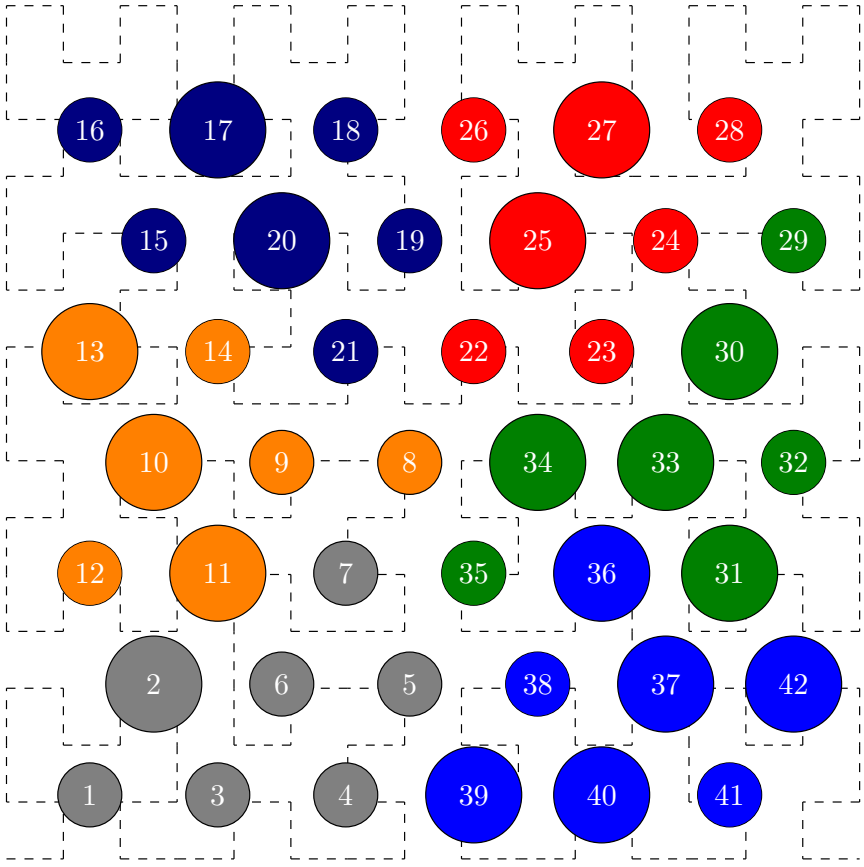
A geração do grafo que define a topologia acontece no início da simulação e repete com uma certa frequência. A estratégia de controle baseada em deslocamento limite, apresentada na subseção 3.4.2, pode ser utilizada com o intuito de disparar os eventos de definição e particionamento do grafo de topologia. Os experimentos numéricos adotados neste trabalho consideram o deslocamento limite igual a $2r_{\max}$.

6.6.3 Particionamento local baseado em HSFC

A estratégia de solução baseada no particionamento local através de HSFC simplifica a definição dos blocos de *threads* através da combinação de tarefas de particionamento e controle de memória. Nesta metodologia de solução o vetor de partículas é ordenado com uma frequência que acompanha a dinâmica das partículas usando o método de controle de memória HSFC (subseção 4.3.2). A definição dos blocos de *threads* é realizada selecionando grupos contíguos de partículas no vetor que contém estes elementos discretos. Este processo herda a característica de localidade espacial das curvas de Hilbert e elimina a constante ordenação dos dados de partículas. Na verdade, estas operações são apenas necessárias durante a atividade de controle de memória.

O particionamento local deste método consiste de duas tarefas: determinação de carga de trabalho e divisão ponderada do vetor de partículas. Estas tarefas são realizadas para cada passo de tempo na simulação, mas são simples de serem realizadas e demandam baixos tempos de processamento. Na tarefa de determinação de carga de trabalho, cada *thread* contabiliza o número de testes geométricos para as suas correspondentes partículas. Depois do cálculo de cada passo de tempo, um indicador de carga de trabalho pode ser obtido a partir do número total de testes geométricos. A carga de trabalho esperada por *thread* pode ser então definida considerando o número de *threads* empregadas e uma divisão homogênea. Este procedimento iterativamente balanceia a carga de trabalho do algoritmo de busca de contatos. Ao contrário das soluções baseadas em RCB e METIS, o balanço de carga de trabalho no método HSFC é realizado para todos os passos de tempo. A Figura 57 ilustra um particionamento local hipotético utilizando seis *threads* (cores) e o método baseado em HSFC.

Figura 57 – Particionamento local usando HSFC.



7 Paralelização distribuída por faixas

7.1 Paralelização baseada em faixas verticais

As análises apresentadas a seguir tem como objetivo avaliar a estratégia de particionamento de domínio baseada em faixas geométricas. A performance do código computacional DEMOOP foi avaliada para tanto em alguns cenários hipotéticos de processamento. O *cluster* da GradeBR/LCCV foi utilizado nos testes realizados. Esta máquina possui 214 processadores Intel i7 (Xeon(R) CPU X5570) cada um com 8 núcleos de processamento e 24 GB de memória RAM.

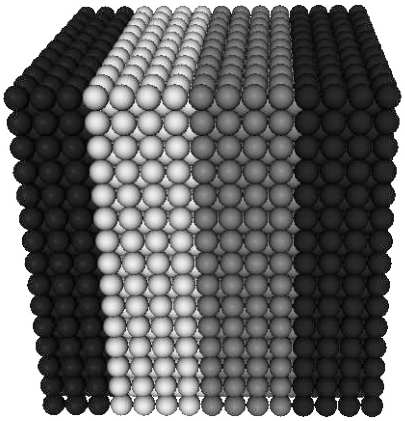
Nos testes de performance realizados, utilizam-se modelos numéricos com descrições geométricas simples e variam-se alguns parâmetros com influência no seu tempo de processamento, tais como o número de partículas e a quantidade de processadores. O problema físico tratado nos modelos é o de acomodação de partículas por gravidade, e a configuração inicial adotada é o arranjo BCC. A geometria do problema é diferenciada em dois cenários de análise visando avaliar a estratégia de particionamento de tarefas adotada na técnica de paralelização. Em um cenário assume-se que o arranjo de partículas está contido em um cubo e no outro o arranjo é gerado no interior de um paralelepípedo com uma dimensão dominante, conforme ilustra a Figura 58. Os tons de cores apresentados ilustram ainda o particionamento do domínio da estratégia de paralelização adotada. No caso ilustrado, por exemplo, utilizam-se quatro processadores (um para cada tom de cor).

Uma vez que sua implementação computacional do DEMOOP baseia-se no padrão de comunicação entre processos via troca de mensagens (padrão MPI), o processamento numérico do sistema pode ser feito utilizando mais de um núcleo de processamento de uma mesma central de processamento (CPU) ou processadores ligados entre si através de um canal de comunicação de rede. Isso torna possível o processamento do mesmo código computacional na forma serial, paralela compartilhada (computador *multicore*) e distribuída (*clusters*). Além disso, é possível ainda a execução do sistema em *clusters* onde as CPU ligadas em rede dispõem de mais de um núcleo de processamento.

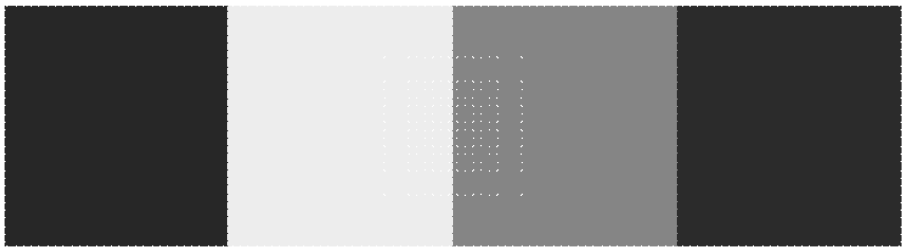
As métricas de comparação utilizadas nas análises são o tempo de processamento, o tempo de comunicação entre processos, *speedup* e eficiência. Entende-se como tempo de processamento o intervalo de tempo total gasto por um conjunto de processadores para a execução coletiva de uma tarefa (simulação de acomodação). O tempo de comunicação entre processos é o tempo total gasto durante o envio de mensagens. *Speedup* é a relação entre o tempo de processamento serial e paralelo de uma dada tarefa, na forma:

$$S(p) = \frac{t_s}{t_p} \tag{7.1}$$

Figura 58 – Modelos geométricos utilizados nas análises.



(a) Cenário cúbico (proporção: $1 \times 1 \times 1$)



(b) Cenário de dimensão dominante
(proporção: $740 \times 200 \times 8$)

onde p é o número de processadores, t_s é o tempo de processamento serial e t_p é o tempo de processamento paralelo para p processadores. A eficiência mede a proporção da capacidade computacional teórica máxima utilizada, e é dada por:

$$e(p) = \frac{t_s}{p\ t_p}, \tag{7.2}$$

que no caso ideal tende à unidade, quando $S(p) = p$, ou seja quando o *speedup* é igual ao número de processadores.

Na Figura 59, apresenta-se a influência do número de processos e do número de partículas (n_p) no tempo computacional para o cenário de geometria cúbica. Os dados de tempo de processamento apresentados no gráficos foram normalizados pelo número de passos de solução utilizados no algoritmo dinâmico do método ($n_{steps} = 1.000$).

Observe que os tempos de processamento tendem a serem maiores com o aumento do número de partículas e diminuição do número de processadores. Nas Figuras 60 e 61 apresentam-se os resultados de eficiência e *speedup* para a simulação do cenário de geometria cúbica.

As curvas apresentadas de *speedup* possuem tendência de crescimento não-linear sobretudo com o aumento do número de processadores. Além disso, os valores de eficiência apresentam alta taxa de decrescimento com o aumento do número de processadores. Nas Figuras 62 e 63 é possível observar a influência das tarefas de comunicação entre processos

Figura 59 – Tempo de processamento (cenário cúbico).

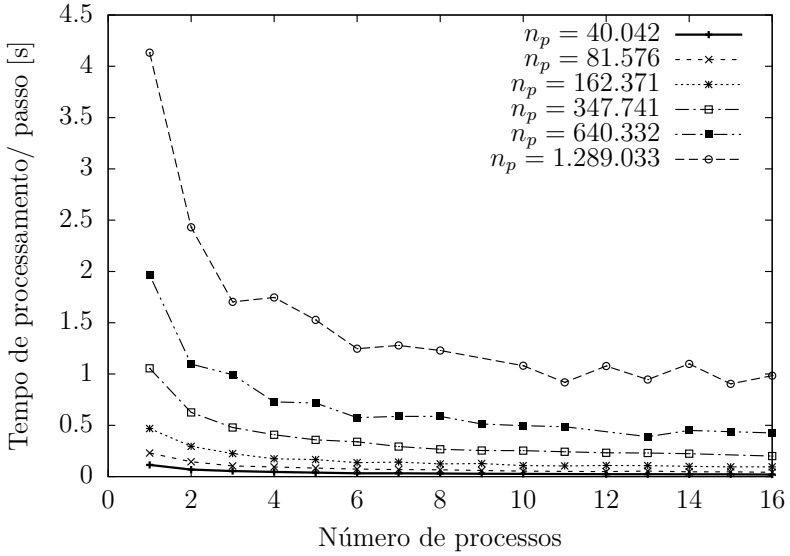
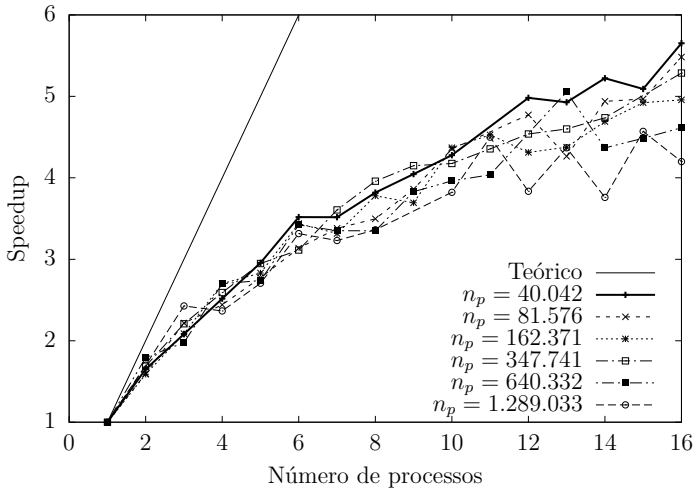


Figura 60 – *Speedup* - $S(p)$ (cenário cúbico).



no tempo total de processamento, em duas escalas distintas: $n_p = 40.042$ (Figura 62) e $n_p = 1.289.033$ (Figura 63). Os dados reportados para a tarefa de comunicação entre processos são apresentadas em curvas distintas, onde uma delas envolve apenas o tempo de comunicação entre processos (rede e overheads) e a outra adicionalmente inclue as tarefas necessárias para envio de mensagens entre processos (*Pack/Unpack* de mensagens).

Os dados de tempos de processamento reportados evidenciam a elevada participação das tarefas de comunicação entre processos. O mesmo ocorre quando a divisão de tarefas de processamento utiliza memória distribuída (1 processo por CPU *multicore*) ou compartilhada (mais de 1 processo por CPU *multicore*) para um problema de pequena escala ($n_p = 40.042$), conforme ilustra a Figura 64.

Isto indica que a carga de tarefas de processamento está decrescendo mais rapidamente que as de comunicação para a escala do problema tratado. É possível observar

Figura 61 – Eficiência - $e(p)$ (cenário cúbico).

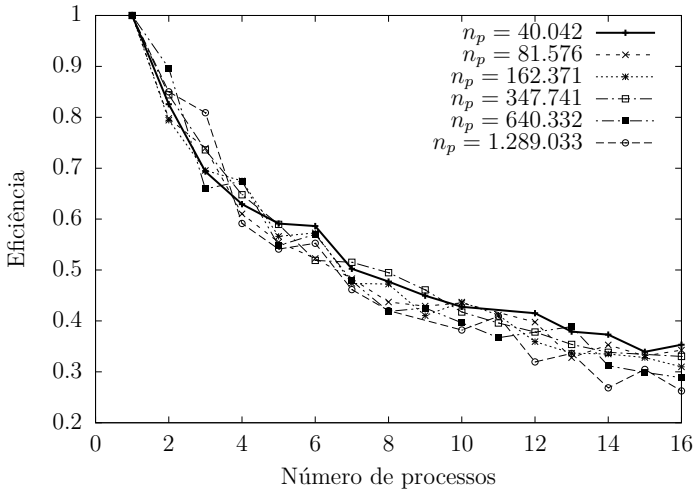
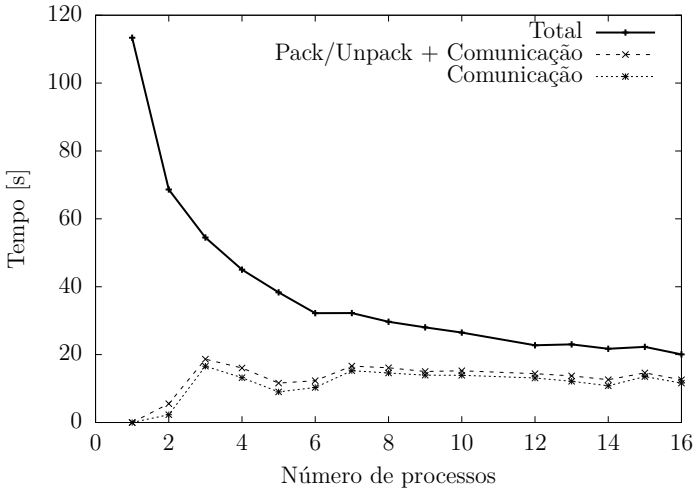


Figura 62 – Influência das tarefas de comunicação no tempo total de processamento - $n_p = 40.042$ (cenário cúbico).



ainda que o tempo de comunicação entre processos de memória distribuída foi ligeiramente superior ao apresentado para memória compartilhada. As Figuras 65 e 66 apresentam, respectivamente, as curvas de *speedup* e eficiência que retratam a influência do modelo de divisão de tarefas.

Note que a estratégia de particionamento de tarefas baseada em divisão do domínio por faixas não é apropriado para o estudo de modelos cuja geometria global tende a um quadrado (no caso bidimensional) ou mesmo um cubo conforme apresentado anteriormente. Para estes casos, a região de comunicação entre processos tende a ser maximizada, acarretando em baixos desempenhos quando no uso de processamento paralelo.

Visando verificar a influência da geometria do modelo na eficiência do código paralelo adotado, reproduz-se um cenário onde uma das dimensões geométricas do modelo é superior às demais. Para tanto adota-se um modelo geométrico de proporções $740 \times 200 \times 8$,

Figura 63 – Influência das tarefas de comunicação no tempo total de processamento - $n_p = 1.289.033$ (cenário cúbico).

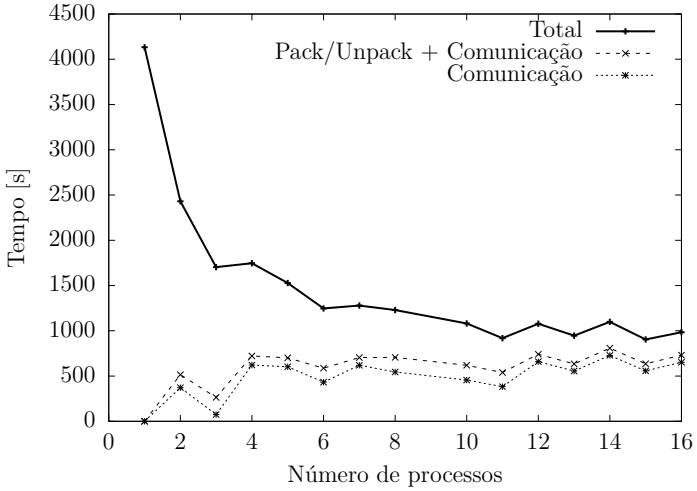
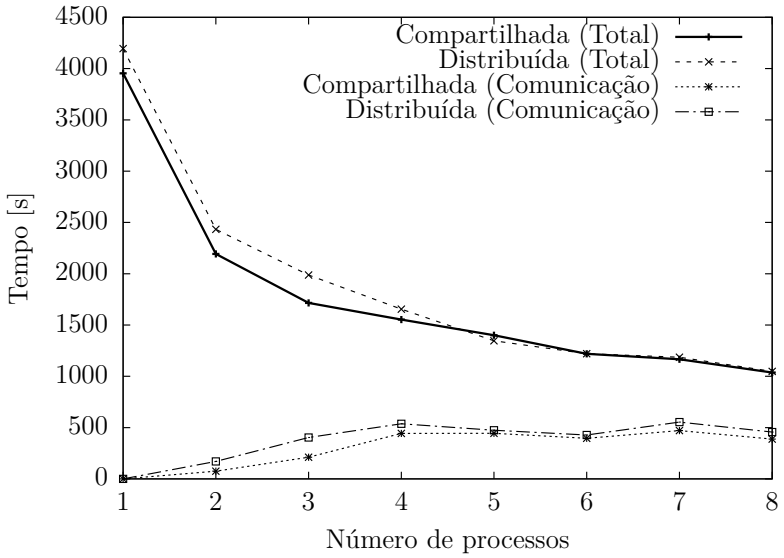


Figura 64 – Influência da forma de divisão de processos entre processadores (cenário cúbico - $n_p = 40.042$).



assim como adotado por Walther e Sbalzarini (2009). Adicionalmente, eleva-se o número de partículas para aumentar a carga de trabalho ($n_p = \{637 \times 10^3;\ 3,23 \times 10^6;\ 6,63 \times 10^6\}$). Neste cenário, a geometria do problema é favorável à metodologia de decomposição por faixas adotada. Além disso, o crescimento do número de partículas faz diminuir a relação entre tempo de comunicação e tempo de processamento.

A Figura 67 apresenta os dados reportados de tempo de processamento para o cenário de dimensão geométrica dominante.

Note que, para este caso, a velocidade de decrescimento da carga de processamento com o aumento do número de processadores é menor. Os dados obtidos para *speedup* e *eficiência*, apresentados nas Figuras 68 e 69 respectivamente, deixam mais evidente tal

Figura 65 – Modelo de divisão de tarefas: *Speedup* - $S(p)$ (cenário cúbico).

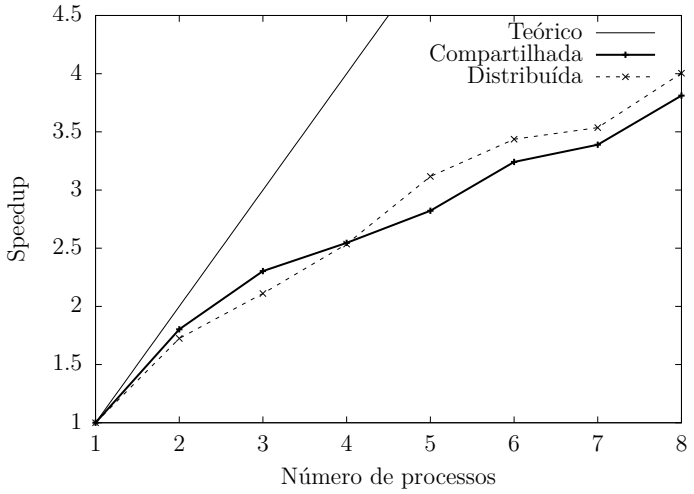
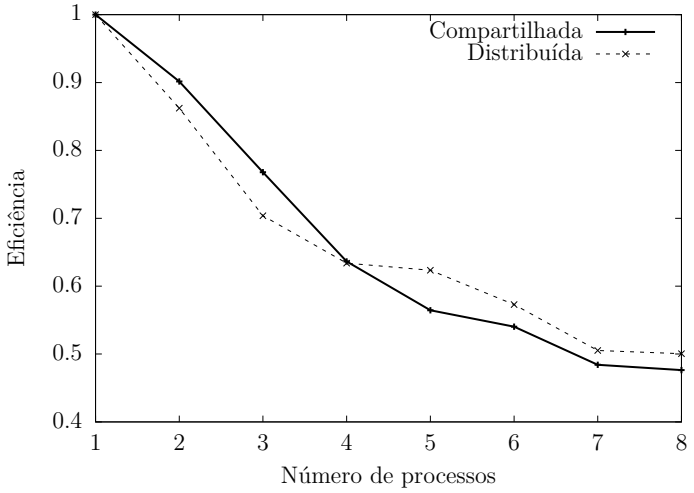


Figura 66 – Modelo de divisão de tarefas: Eficiência - $e(p)$ (cenário cúbico).



fato.

Em modelos de larga escala, mesmo em cenários com dimensões geométricas dominantes, a comunicação é um dos limitantes para o tempo de processamento. A Tabela 3 apresenta os dados de tempo de processamento e comunicação para um modelo de grande porte contendo $123,68$ milhões de partículas (cenário geométrico dominante).

Tabela 3 – Performance em modelo de larga escala ($n_p = 123,68 \times 10^6$)

	192 processadores	256 processadores
Tempo de processamento por passo (total)	$10,4853$s	$11,4116$s
Tempo de solução por passo	$2,6829$s	$2,3852$s
Tempo de comunicação por passo	$7,8024$s	$9,0264$s

Fonte: O autor.

Note que, embora exista uma diminuição do tempo de solução quando o número de processadores aumenta, o tempo de comunicação acaba elevando-se com esta mudança e

Figura 67 – Tempo de processamento (cenário de dimensão dominante).

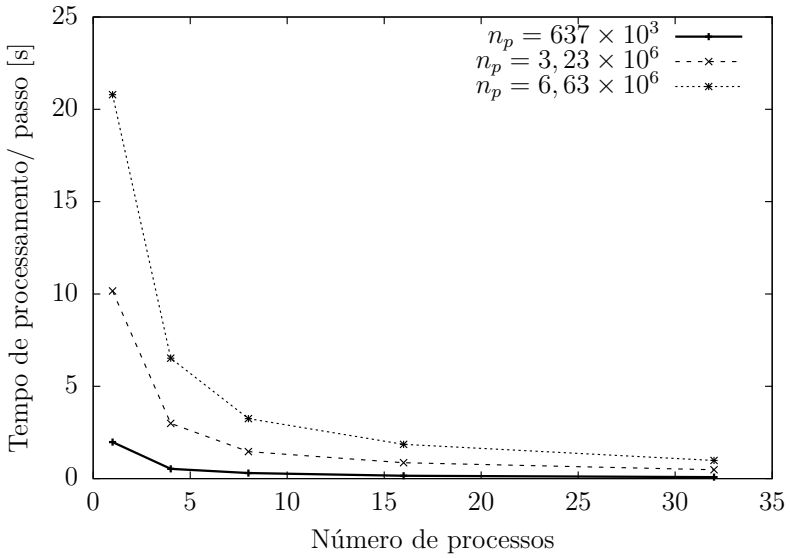
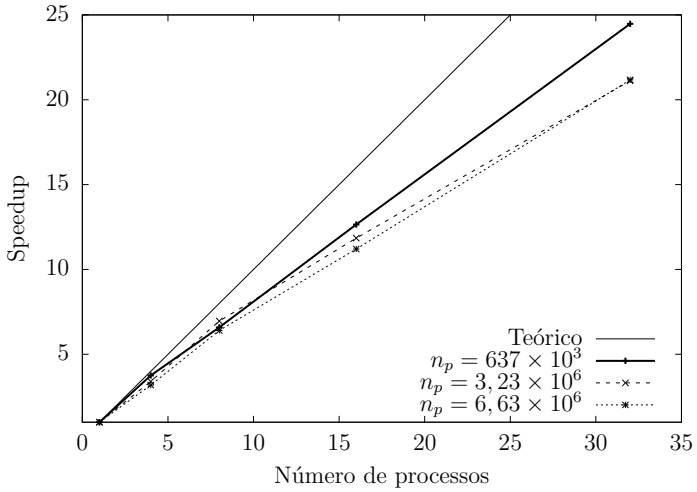
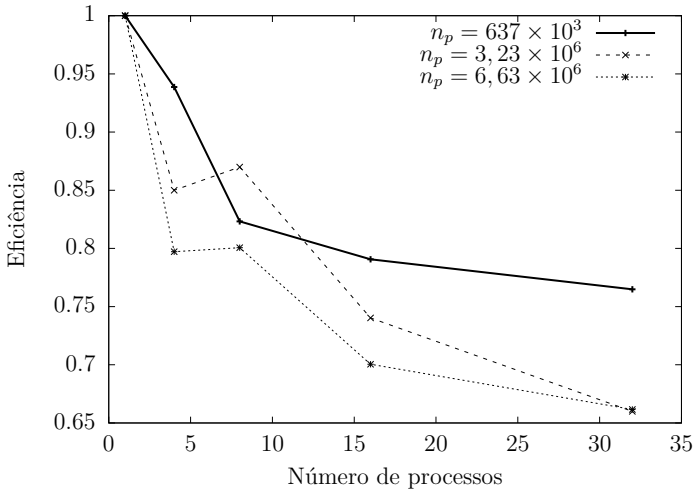


Figura 68 – *Speedup* - $S(p)$ (cenário de dimensão dominante).



interferindo no tempo total de processamento de forma indesejável. O emprego de técnicas que minimizem a tarefa de comunicação entre processos é, portanto, uma exigência para a simulação de modelos com esta magnitude.

Figura 69 – Eficiência - $e(p)$ (cenário de dimensão dominante).



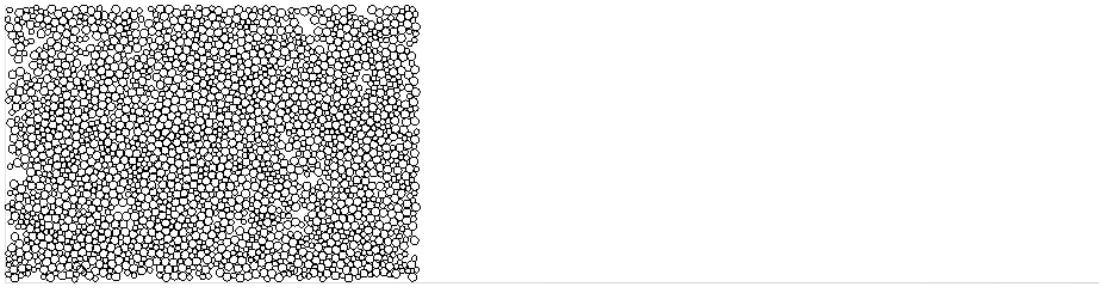
8 Paralelização distribuída - topológica

8.1 Particionamento de domínio e balanceamento de carga

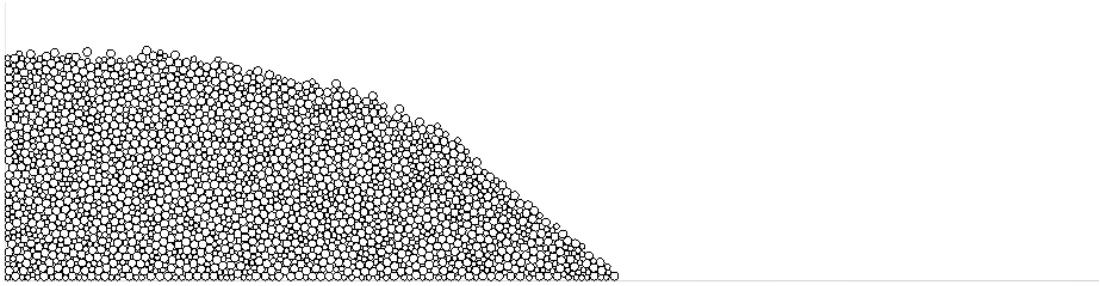
8.1.1 Simulação de quebra de barragem

Visando avaliar as estratégias de particionamento de domínio para simulações por elementos discretos, desenvolve-se, a seguir, um estudo onde algumas técnicas de particionamento de domínio e balanceamento de carga são empregadas. A simulação adotada nas análises é a de um modelo de quebra de barragem, onde o volume deslocado representa uma massa sólida discretizada em partículas. A Figura 70 ilustra o problema de quebra de barragem para dois estágios da simulação, correspondentes à configuração inicial do modelo, Figura 70 (a), e a um instante onde a massa sólida está em movimento, Figura 70 (b). O modelo é composto por um conjunto de 2.202 partículas com configuração inicial gerada através do algoritmo de separação geométrica (subseção 2.2.1.1).

Figura 70 – Cenário hipotético de quebra de barragem.



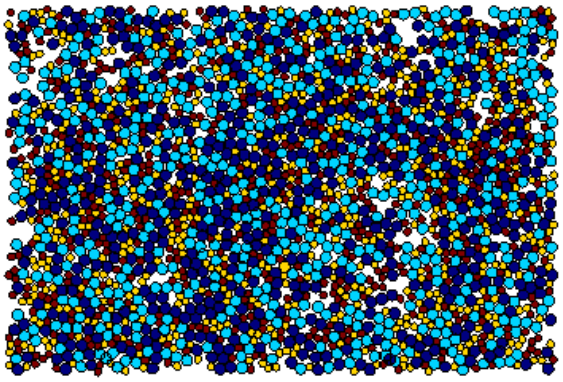
(a) Configuração inicial



(b) Movimento

O problema consiste em particionar o domínio referente ao conjunto de partículas em 4 subdomínios. A Figura 71 apresenta a distribuição inicial, aleatória, adotada para posterior particionamento. Os subdomínios são apresentados em cores distintas correspondentes.

Figura 71 – Distribuição inicial de partículas nos processos.



Observe que, neste caso, existe um número grande de interfaces entre subdomínios uma vez que não há um agrupamento espacial das partículas por subdomínio. Isso ocorre devido à aleatoriedade do processo de geração de partículas adotado.

As estratégias de particionamento geométrico, topológico e híbrido são utilizadas com o intuito de avaliar a qualidade da decomposição resultante. Na técnica de particionamento híbrido, as informações geométricas são utilizadas para acelerar a solução do particionamento topológico multi-nível a ser realizado posteriormente. Os particionamentos são apresentados em instantes distintos da simulação, indicando estágios de grande movimentação das partículas, de dissipação de energia, e de repouso. Neste trabalho o *software* utilizado para o particionamento de domínio é o ParMETIS (KARYPIS; KUMAR, 1998b).

Em sequência, na subseção 8.1.1.4, emprega-se uma técnica de balanceamento dinâmico de carga para avaliar a evolução de um particionamento dado ao longo tempo. O *software* utilizado para balanceamento dinâmico de cargas é o Zoltan (BOMAN et al., 2012).

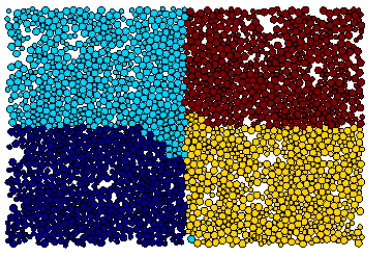
Em ambos os estudos de particionamento de domínio e balanceamento dinâmico de cargas, utilizam-se dados temporais de simulações executadas previamente na forma serial. Para a execução dos particionamentos e balanceamentos, foram utilizados exatos 4 processos *MPI*.

8.1.1.1 Particionamento geométrico

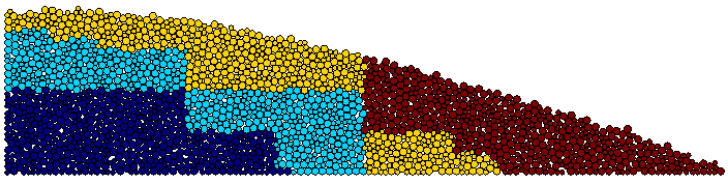
Na técnica de particionamento geométrico apresentada, o algoritmo adotado para a decomposição de domínio é baseado em coordenadas e utiliza o método de curvas de preenchimento de espaço (subseção 3.1.4).

A Figura 72 apresenta os particionamentos geométricos obtidos para os estágios inicial t_0, de dissipação de energia (t_1) e de repouso (t_f).

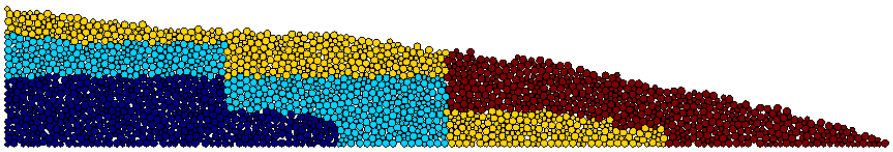
Figura 72 – Particionamento geométrico.



(a) $t_0 = 0$ s



(b) $t_1 = 7$ s



(c) $t_f = 14$ s

Observa-se que os particionamentos obtidos geram subdomínios com aglomerações de partículas bem definidas. Verifica-se ainda que as interfaces entre subdomínios são preferencialmente ortogonais aos eixos cartesianos e de grande extensão.

8.1.1.2 Particionamento de grafos em multi-nível

Na técnica de particionamento topológico, definem-se nós e arestas para a representação da topologia do problema e busca-se a definição de subdomínios de forma a minimizar o número de cortes de arestas da topologia. Esta minimização no entanto deve ser feita de forma a balancear o número de nós para os subdomínios.

O critério de definição das arestas deve refletir as interações necessárias entre os nós. No caso particular de modelos de elementos discretos estas arestas podem representar por exemplo o mapa de contato entre partículas ou mesmo um diagrama de vizinhança. Os grafos adotados para definição de topologia usando os critérios de contato e de vizinhança, em um estágio intermediário da simulação, são ilustrados na Figura 73 e na Figura 74 respectivamente. O critério de contato assume a existência de uma aresta no grafo quando existe uma sobreposição geométrica entre duas partículas do modelo (subseção 2.4.2). Para

o critério baseado em vizinhança, assume-se a existência de uma aresta no grafo quando a distância entre os centros das partículas é menor que cinco diâmetros.

Figura 73 – Grafo para critério de contato.

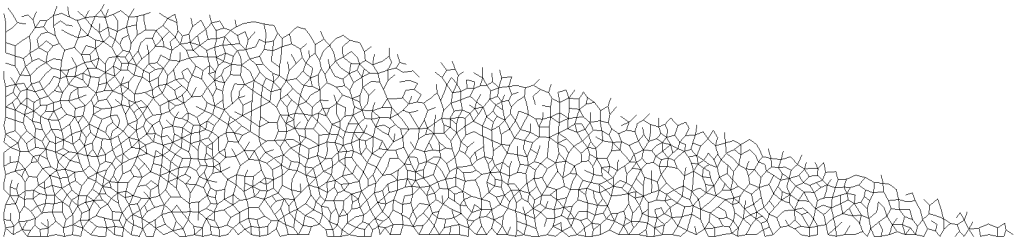
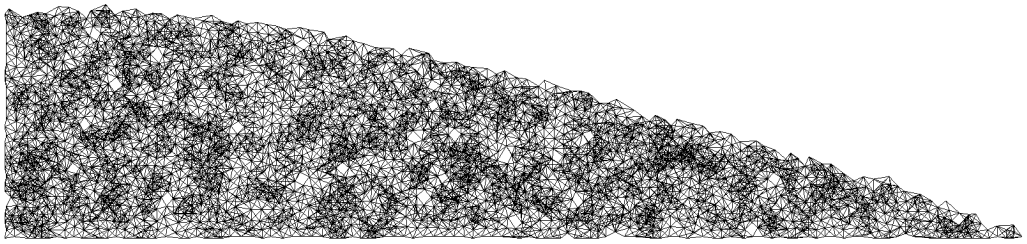
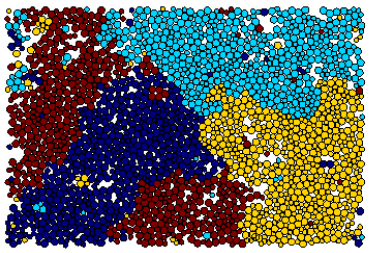


Figura 74 – Grafo para critério de vizinhança.

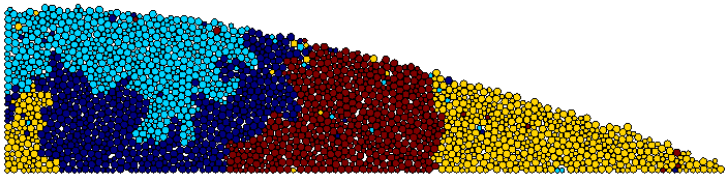


As Figuras 75 e 76 ilustram o emprego destes dois critérios para o particionamento topológico. São reportados particionamentos distintos para diferentes instantes de tempo na simulação fictícia de quebra de barragem. Na Figura 75 emprega-se o critério de contato, enquanto que na Figura 76 utiliza-se o critério de vizinhança.

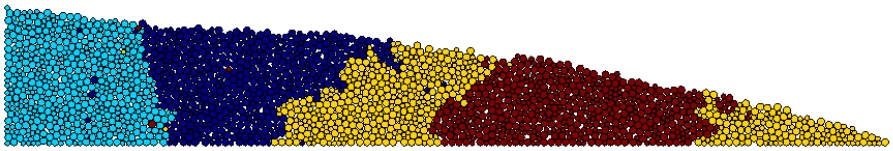
Figura 75 – Particionamento topológico (contato).



(a) $t_0 = 0$ s



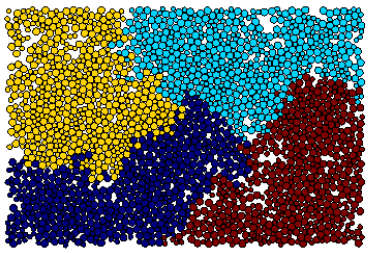
(b) $t_1 = 7$ s



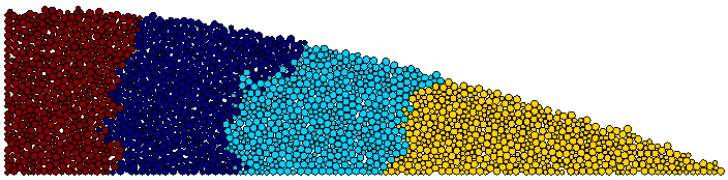
(c) $t_f = 14$ s

Observa-se na Figura 75, para os instantes de tempo apresentados, a ocorrência de partículas de cores distintas em regiões distantes da pseudo-fronteira de subdomínios. O mesmo comportamento não é, geralmente, verificado quando o critério de vizinhança é adotado (Figura 76).

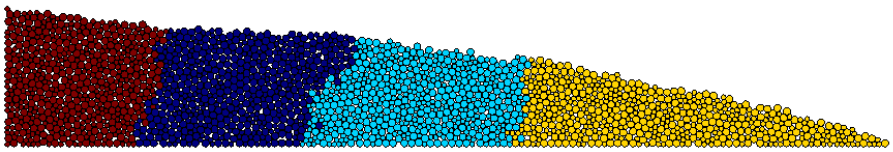
Figura 76 – Particionamento topológico (vizinhança).



(a) $t_0 = 0$ s



(b) $t_1 = 7$ s



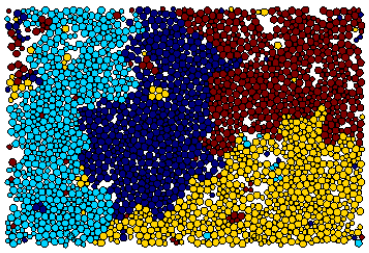
(c) $t_f = 14$ s

8.1.1.3 Particionamento híbrido (geométrico e topológico)

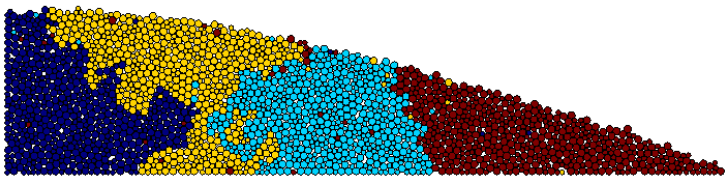
Na técnica de particionamento híbrido (geométrico e topológico), ambas as informações de posição espacial e topologia são utilizadas na decomposição do domínio.

As Figuras 77 e 78 apresentam os particionamentos híbridos obtidos para os critérios de contato (Figura 77) e vizinhança (Figura 78). Novamente, o particionamento é feito para 4 processos e, quando o critério de vizinhança é adotado, define-se uma aresta da topologia quando a distância entre partículas é menor que cinco diâmetros.

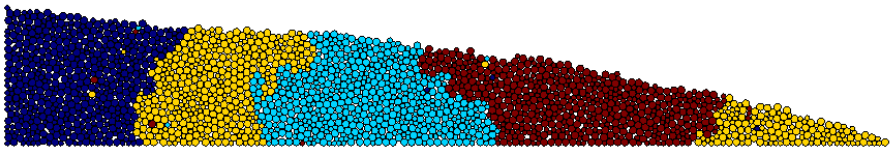
Figura 77 – Particionamento híbrido (contato).



(a) $t_0 = 0$ s



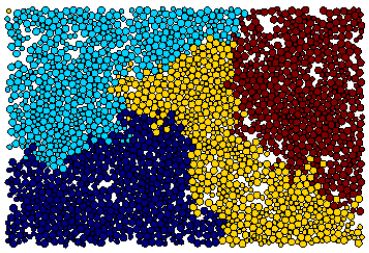
(b) $t_1 = 7$ s



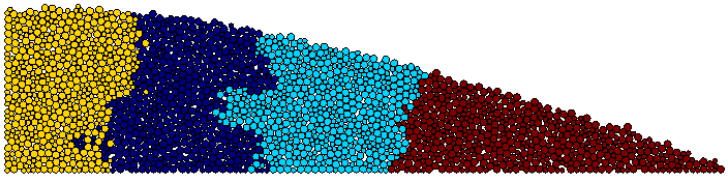
(c) $t_f = 14$ s

Observa-se pouca alteração para o modelo apresentado, quando a estratégia combinada é utilizada em relação ao caso onde se utiliza apenas a topologia (Figuras 75 e 76).

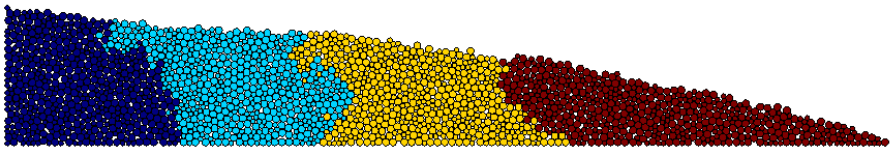
Figura 78 – Particionamento híbrido (vizinhança).



(a) $t_0 = 0$ s



(b) $t_1 = 7$ s



(c) $t_f = 14$ s

8.1.1.4 Balanceamento dinâmico de carga

No procedimento de balanceamento de carga adotado, o particionamento inicial apresentado na Figura 71 é decomposto em 4 subdomínios utilizando a metodologia de particionamento multi-nível para o critério de vizinhança. Em seguida, geram-se decomposições consecutivas que são obtidas através de uma metodologia baseada em migração de nós entre subdomínios. Estas decomposições são realizados ao longo do tempo de simulação, para diversos instantes de tempo, à medida que as partículas se movimentam no domínio do problema estudado. Os dados de decomposição apresentados nesta seção apenas ilustram alguns instante de tempo correspondentes a estágios de tempo de interesse.

Na estratégia de decomposição de domínio empregada tenta-se minimizar uma função multi-objetivo que considera o número de cortes de arestas e o volume de migração de nós (partículas) entre subdomínios, na forma:

$$f_{obj}(\alpha) = \alpha(\text{cortes de arestas}) + (\text{volume de migração}), \tag{8.1}$$

onde α é um multiplicador que pode ser selecionado de maneira a aumentar ou diminuir a importância dada ao corte de arestas em relação ao volume de migração. Maiores detalhes

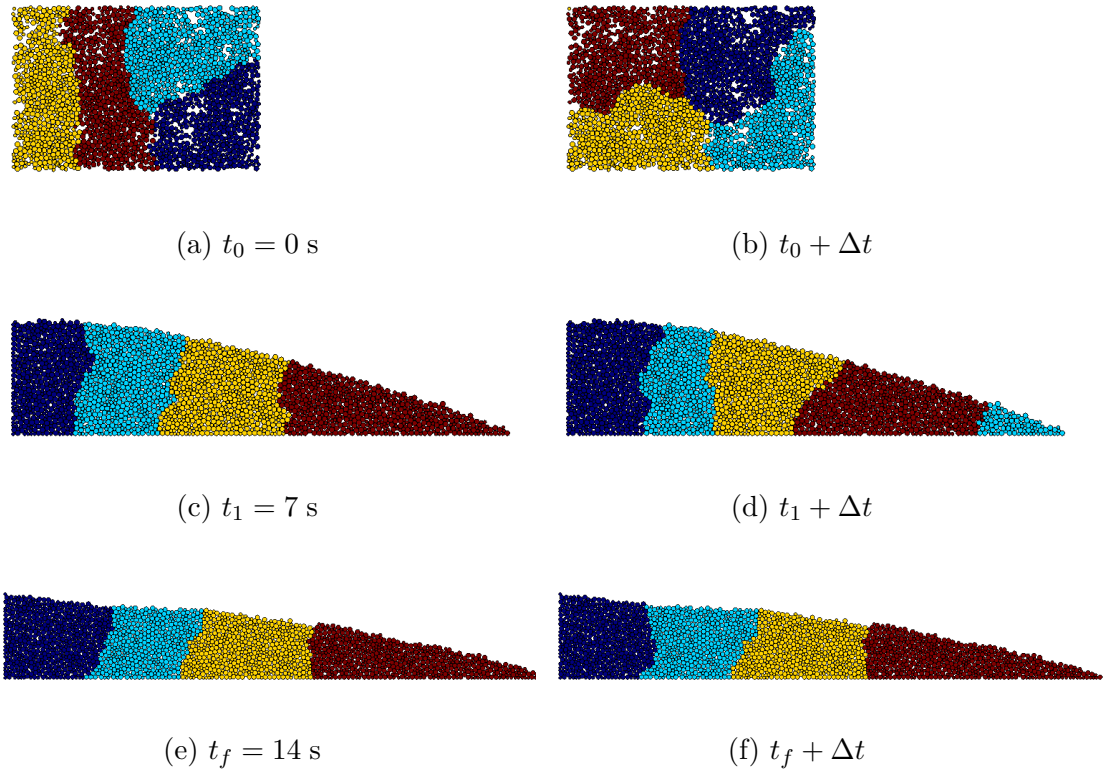
sobre o multiplicador estão disponíveis no manual de usuário da biblioteca Zoltan (BOMAN et al., 2007).

Em geral, quando a redução do número de cortes de aresta é priorizada a qualidade do particionamento é melhor. No entanto, isso acaba elevando o volume de migração, que ocorre quando uma partícula deixa de pertencer a um subdomínio para pertencer a outro em um passo de tempo posterior. Quando se prioriza a redução de migração, os nós tendem a permanecer nos subdomínios e cortes de arestas acabam surgindo como consequência. Isto ocorre devido aos deslocamentos dos vértices do grafo e a consequente geração de novas arestas.

Visando avaliar a influência do parâmetro que controla a função objetivo (Equação 8.1) no balanceamento de carga empregado, apresenta-se a seguir um estudo paramétrico onde o multiplicador α é investigado. São reproduzidos quatro cenários distintos, reportando situações limites e cenários balanceados de uso da variável α. As decomposições são apresentadas, para cada valor de α adotado, em três estágios distintos (t_0,t_1 e t_f). Para cada um destes estágios apresenta-se ainda seu correspondente particionamento consecutivo, obtido após um passo de integração temporal de tamanho Δt.

No primeiro cenário prioriza-se a redução do número de cortes de arestas. Para tanto, utiliza-se o valor de $\alpha = 100$. A Figura 79 apresenta os particionamentos para esta configuração.

Figura 79 – Balanceamento dinâmico de carga - $\alpha = 100$.



(a) $t_0 = 0$ s (b) $t_0 + \Delta t$

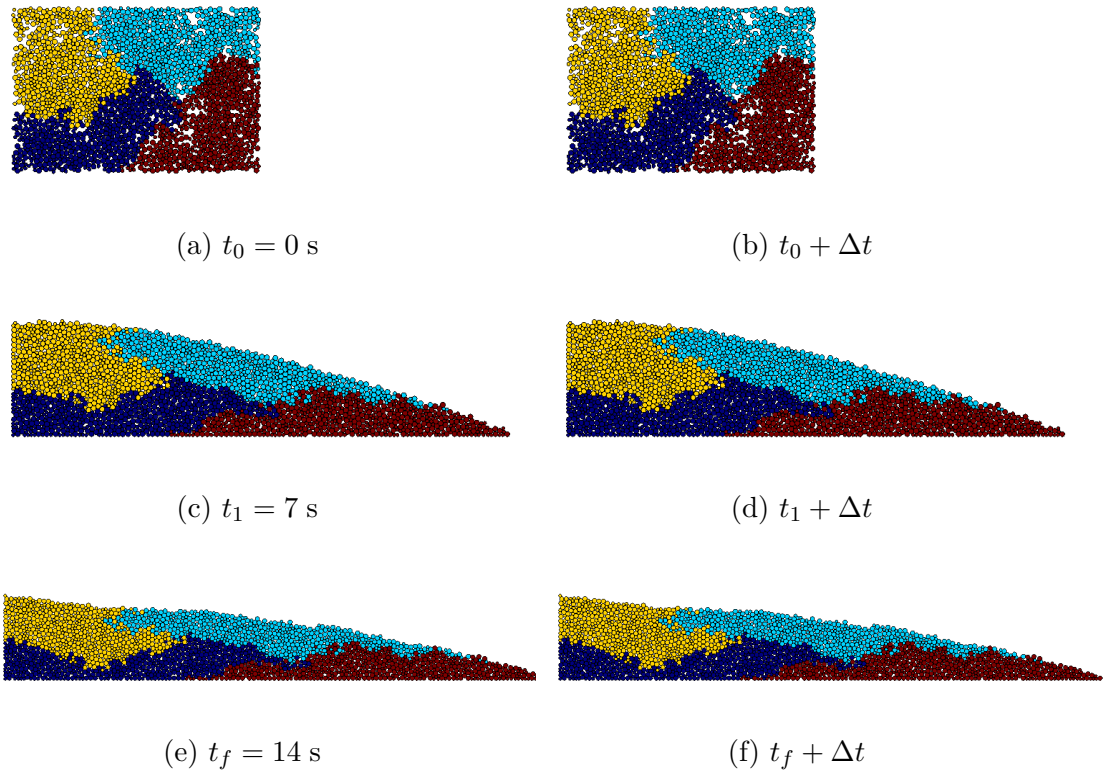
(c) $t_1 = 7$ s (d) $t_1 + \Delta t$

(e) $t_f = 14$ s (f) $t_f + \Delta t$

Observe que, neste caso, ocorre uma variação significativa dos subdomínios ao longo do tempo. Isto fica bem visível no estágio intermediário t_1, onde seu correspondente particionamento consecutivo, $t_1 + \Delta t$, tem configuração de subdomínios bastante distinta.

Em um cenário limite pode-se priorizar a redução do volume de migração entre subdomínios utilizando o valor de $\alpha = 0$. Neste cenário, o que ocorre é que não se dá importância ao corte de arestas na definição da função objetivo. Logo, os particionamentos consecutivos ocorrem de maneira a tentar manter a configuração inicial de particionamento. Como na configuração inicial o particionamento já está balanceado, então não há migração de partículas entre subdomínios. As partículas apenas deslocam-se e modificam a forma apresentada para os subdomínios, conforme ilustra a Figura 80.

Figura 80 – Balanceamento dinâmico de carga - $\alpha = 0$.



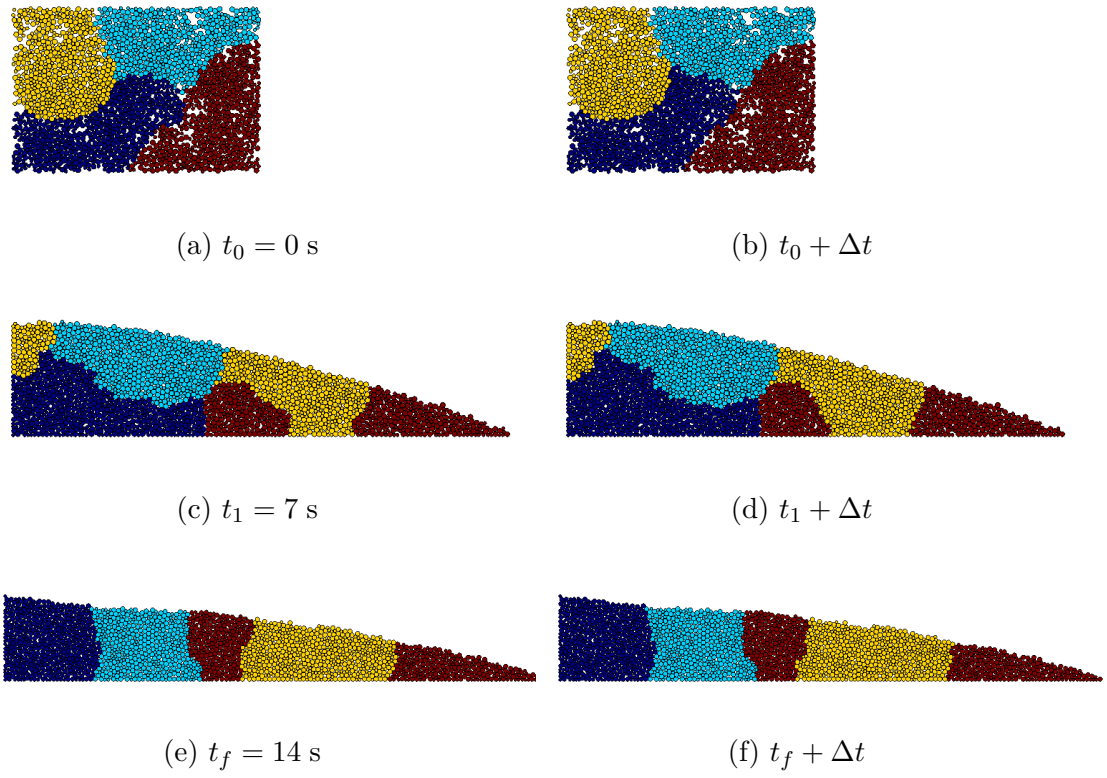
(a) $t_0 = 0$ s (b) $t_0 + \Delta t$

(c) $t_1 = 7$ s (d) $t_1 + \Delta t$

(e) $t_f = 14$ s (f) $t_f + \Delta t$

Na Figura 81, apresentam-se os particionamentos obtidos quando a variável α assume o valor unitário. Isso indica que a mesma importância, tanto para redução de cortes de arestas como para o volume de migração, é adotada.

Figura 81 – Balanceamento dinâmico de carga - $\alpha = 1$.



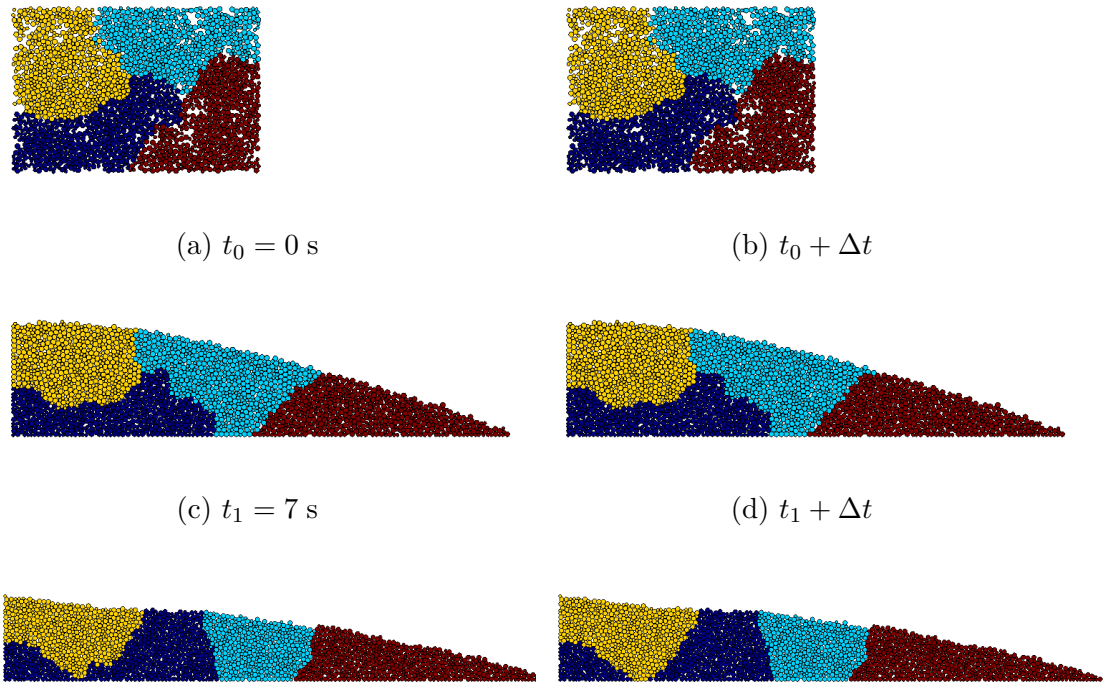
(a) $t_0 = 0$ s (b) $t_0 + \Delta t$

(c) $t_1 = 7$ s (d) $t_1 + \Delta t$

(e) $t_f = 14$ s (f) $t_f + \Delta t$

Nesta configuração as partículas migram de um subdomínio para o outro, no entanto ainda se observa no estágio t_f uma variação de decomposição para particionamentos consecutivos. Isto pode ser resolvido com um ajuste fino da variável α. Na Figura 82, o valor de $\alpha = 0,5$ foi utilizado com este propósito.

Figura 82 – Balanceamento dinâmico de carga - $\alpha = 0,5$.



(a) $t_0 = 0$ s (b) $t_0 + \Delta t$

(c) $t_1 = 7$ s (d) $t_1 + \Delta t$

(e) $t_f = 14$ s (f) $t_f + \Delta t$

8.2 Corrida de detritos 2D

Os movimentos de massa de terra ou avalanches são eventos raramente monitorados em campo em virtude da característica de movimento rápido e falha súbita. Em muitos casos estes eventos são motivos de grandes perdas materiais e humanas, sendo preocupações frequentes de autoridades públicas e privadas.

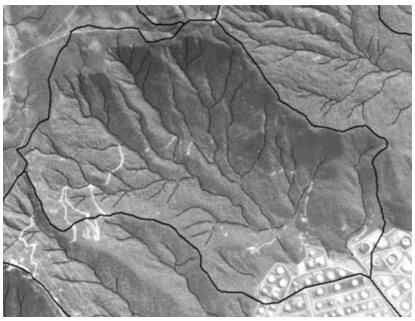
Grande parte das instalações da Petrobras, por exemplo, estão situadas na região centro-sul do país, região essa margeada pela Serra do Mar. Esta região é densamente povoada, possui relevo bastante acidentado e condicionantes geológicos e hidrológicos que propiciam instabilidades de taludes.

Tipicamente, na Serra do Mar ocorrem movimentos lentos (rastejos) ou rápidos (corridas). As corridas de detritos/lama observadas são predominantemente translacionais com pequenas profundidades e ocorrem em perfis de solo residuais e/ou em colúvios sobre maciço rochoso. Anualmente, vários eventos desse tipo afetam a infraestrutura da Petrobras. São observados casos com graves consequências com uma frequência relativamente alta.

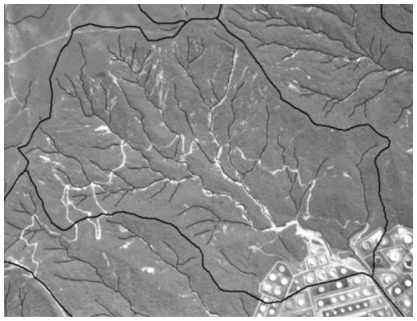
No evento que afetou a bacia do Rio das Pedras, e consequentemente a refinaria Presidente Bernardes em Cubatão, o volume total de detritos deslocado foi estimado em 300.000 m^3 de lama, areia, pedregulhos, rocha e troncos de árvores (LOPES; RIEDEL, 2007). A velocidade média do fluxo de detritos, estimada por meio de fórmulas empíricas

(MASSAD et al., 1999), foi da ordem de 10 m/s. O prejuízo calculado com limpeza, reparos e 3 semanas de interrupção de operação na refinaria foi da ordem de 40 milhões de dólares (KANJI; CRUZ; MASSAD, 2008). A Figura 83 apresenta as ortofotos da Bacia do Rio das Pedras antes e após o evento de corrida de detritos, onde pode-se observar a erosão no terreno ocasionada pelo evento.

Figura 83 – Comparação entre as ortofotos de 1986 e 1994 (Bacia do Rio das Pedras).



(a) 1986 - Antes do evento.



(b) 1994 - Após a corrida de detritos.

Ao tratar os estudos que envolvem essa problemática, na maioria dos casos, utilizam-se métodos empíricos, formulações de equilíbrio limite para cálculo de estabilidade de taludes ou mesmo análises numéricas baseadas no Método dos Elementos Finitos. Neste trabalho, alternativamente, utiliza-se o Método dos Elementos Discretos. A escolha baseia-se no seu potencial de aplicabilidade para problemas de natureza descontínua em especial referentes a corridas de detritos (CAMPBELL; CLEARY; HOPKINS, 1995; TANG et al., 2009; MUNJIZA; CLEARY, 2009).

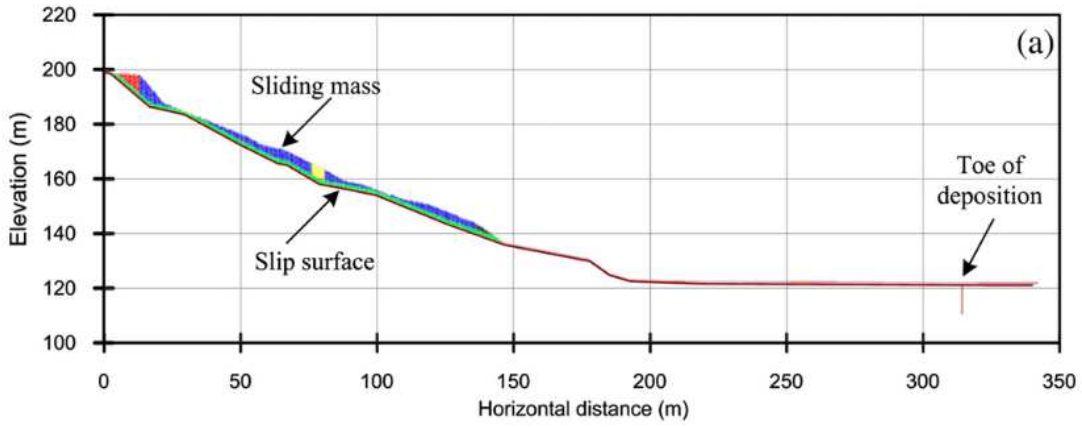
Pretende-se avaliar o desempenho das técnicas de paralelização desenvolvidas em um cenário de corrida previamente analisado na literatura (LI et al., 2012). Para tanto adota-se a corrida de detritos ocorrida no colina de Yangbaodi, na cidade de Shenzhen, sudeste da China (Figura 84). O incidente ocorreu em setembro de 2002 após chuvas intensas prolongadas e mobilizou um volume de solo de 25.000 m^3. O alcance da corrida foi de 140 metros, atingiu altas velocidades de fluxo e acarretou perdas humanas.

Figura 84 – Corrida de detritos de Yangbaodi.

Fonte: Li et al. (2012).

Li et al. (2012) apresentam um modelo numérico baseado no DEM para o estudo da corrida de Yangbaodi. No trabalho apresentado, estuda-se a influência de parâmetros microscópicos relacionados com o atrito entre a partículas e o topo rochoso em dados macroscópicos da corrida, tais como o alcance, históricos de velocidades e perfis de altura do solo em sua configuração final. A simulação realizada emprega o *software* PFC2D (GROUP, 2014) e uma discretização que considera 3.022 partículas com diâmetro de $0,2$ m, conforme ilustra a Figura 85.

Figura 85 – Modelo numérico de Li et al. (2012).



Fonte: Li et al. (2012).

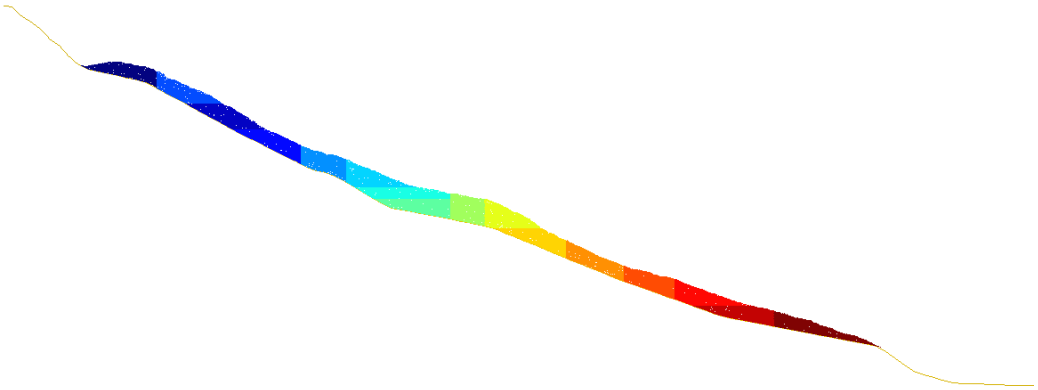
Em uma modelagem de fluxo o tamanho das partículas deve ser adequadamente pequeno para simular os movimentos relativos das partículas nas direções vertical e longitudinal (LI et al., 2012). Observa-se, no entanto, que pequenas alterações nas dimensões

das partículas acabam elevando rapidamente o tamanho do modelo numérico e consequentemente o tempo de processamento demandado. Para tornar viável a análise de simulações de corrida como as descritas anteriormente faz-se necessário o emprego de técnicas de paralelização.

8.2.1 Descrição do cenário simulado

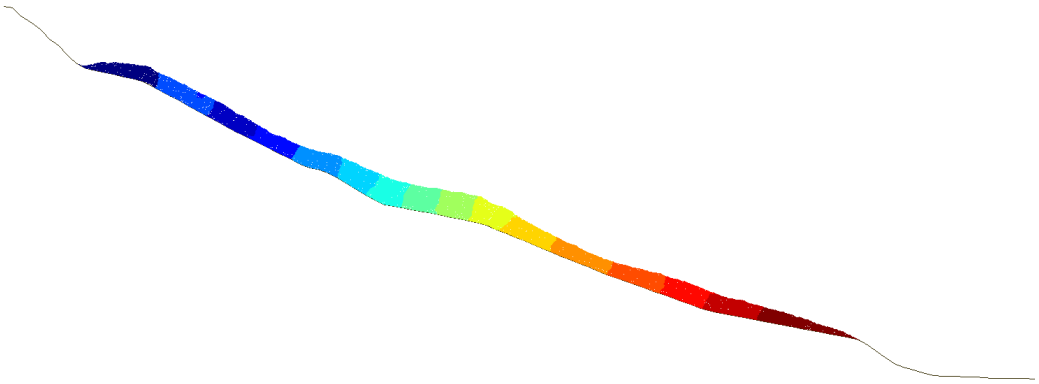
No modelo numérico adotado neste trabalho, a corrida de Yangbaodi é reproduzida através do DEM para uma discretização mais apurada. Adota-se para tanto um conjunto com 272.573 partículas (diâmetro de $0,05$ m). Investiga-se então a eficiência de técnicas de paralelização baseadas no particionamento geométrico (*RCB*), em topologia (definida por vizinhança) e na técnica combinada baseada em redirecionamento baseado em células (seção 5.5). Os particionamentos obtidos nos cenários avaliados seguem padrões típicos. Na técnica de particionamento geométrico (*RCB*), as divisões de domínio são impostas ortogonais aos eixos cartesianos, conforme ilustra a Figura 86.

Figura 86 – Particionamento para 16 processos - *RCB*.



Nas técnicas de particionamento baseadas em topologia, não existe uma imposição de corte para definição dos subdomínios. Verifica-se no entanto que as partículas do mesmo processo, ilustradas com a mesma cor, estão aglomeradas espacialmente conforme apresenta a Figura 87.

Figura 87 – Particionamento para 16 processos - Topologia.



O detalhe aproximado dos particionamentos obtidos na região central do domínio é apresentado na Figura 88. Nesta ilustração fica mais evidente os padrões de particionamentos envolvidos. É possível observar ainda as interfaces entre processos com maior detalhe.

Figura 88 – Detalhe do particionamento.

(a) Topologia. (b) RCB.

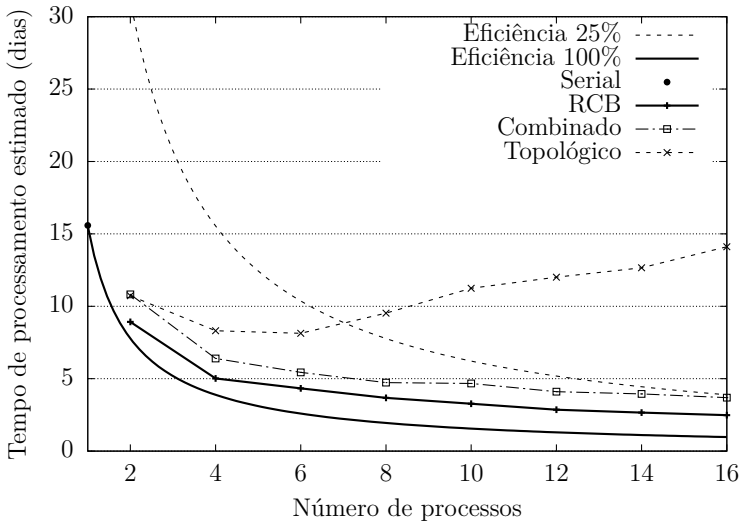
Note que o particionamento geométrico gera interfaces entre processos com comprimentos mais variáveis que o observado na técnica topológica. No caso específico ilustrado na Figura 88(b), um dos subdomínios possui mais de dois processos vizinhos com duas de suas interfaces muito alongadas, o que acarreta em aumento da comunicação entre processos.

8.2.2 Métricas de desempenho

As simulações são realizadas para conjuntos de processadores envolvendo até 16 processos (foram utilizados dois nós com oito núcleos de processamento cada). As métricas adotadas no estudo envolvem dados referentes aos tempos de processamento do modelo, balanços de carga e volume de comunicação entre processos. Para as medições de desempenho adota-se uma simulação truncada do modelo para os primeiros 25000 passos de integração temporal, o que corresponde a 1% do total necessário para a simulação completa.

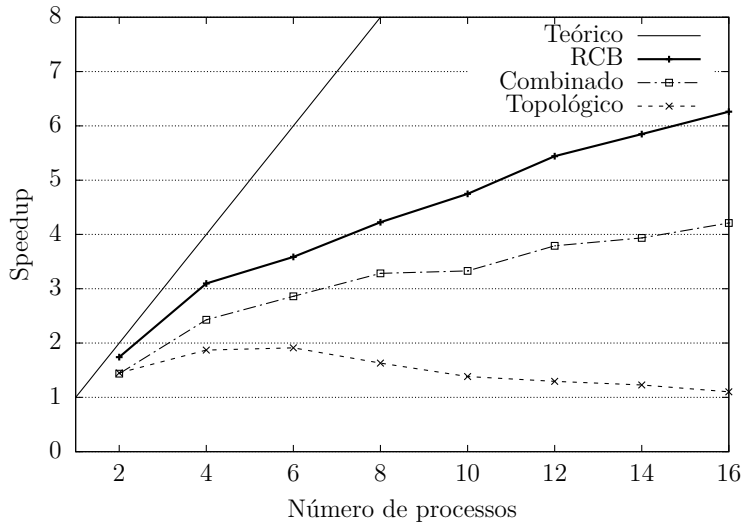
A Figura 89 reporta os tempos de processamento estimados para as técnicas de paralelização adotadas. Os dados de tempo de processamento são apresentados tomando como base duas curvas de desempenhos teóricos correspondentes a 100% e 25% de eficiência, retratando cenários de ótimo e baixo desempenho, respectivamente.

Figura 89 – Tempo de processamento estimado para corrida com 272.573 partículas.



Observe que o modelo adotado, quando executado de forma serial, demanda um tempo de processamento superior a 15 dias. Análises com tal porte são proibitivas sobretudo em estudos paramétricos tais como o realizado por Li et al. (2012). Note ainda que a técnica de paralelização baseada em bisseção de coordenadas (*RCB*) e a estratégia combinada alcançam ganhos consideráveis em tempo de processamento. A perda de eficiência da paralelização ocorre de forma gradual com o aumento do número de processos. Na técnica de particionamento topológico, no entanto, observa-se uma redução brusca da eficiência. Note que a partir de 8 processos, a técnica topológica já possui eficiência inferior a 25%. Os dados de eficiência da paralelização também podem ser observados na Figura 90 para as curvas de *speedup*.

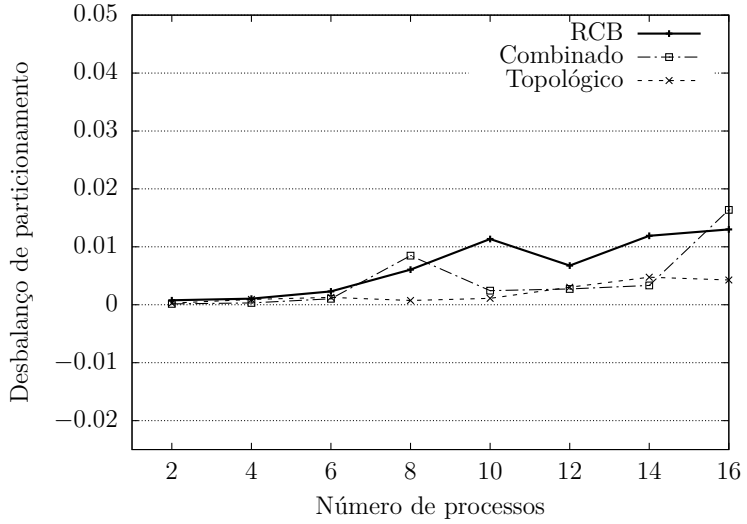
Figura 90 – *Speedup* para corrida com 272.573 partículas.



Observe que a técnica de paralelização baseada em geometria (*RCB*) obteve o melhor desempenho dentre as demais, embora com *speedup* da ordem de 6 para 16 processos.

A Figura 91 apresenta a média temporal do balanço de cargas para as técnicas avaliadas. Os dados apresentados são valores relativos e indicam desvios em torno da média para o número de partículas por processo.

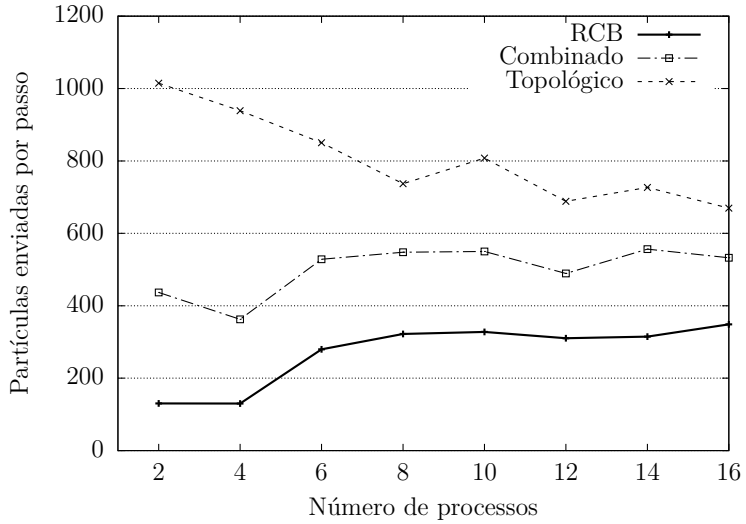
Figura 91 – Desbalanço de particionamento para corrida com 272.573 partículas.



Observa-se que, mesmo nos casos mais críticos, o desbalanço no número de partículas é inferior a 2%, indicando um bom balanço de particionamento. As técnicas de particionamento que consideram a topologia obtiveram menores desbalanços na maioria dos casos. Quando o balanço não é bem atendido para uma simulação, um processo demanda mais tempo que os demais em suas tarefas e torna os demais processos ociosos e menos eficientes.

Visto que o balanço de particionamento não é o fator determinante para a baixa eficiência obtida para as técnicas de paralelização baseadas em topologia, parte-se para o estudo do volume de comunicação entre processos. Com este propósito adota-se como métrica o número médio de partículas enviadas entre processos por passo de integração. Desta forma, quanto menor o número de partículas enviadas, menor é o volume de comunicação e melhor é a eficiência do sistema. A Figura 92 apresenta os dados de volume de comunicação obtidos como função do número de processos.

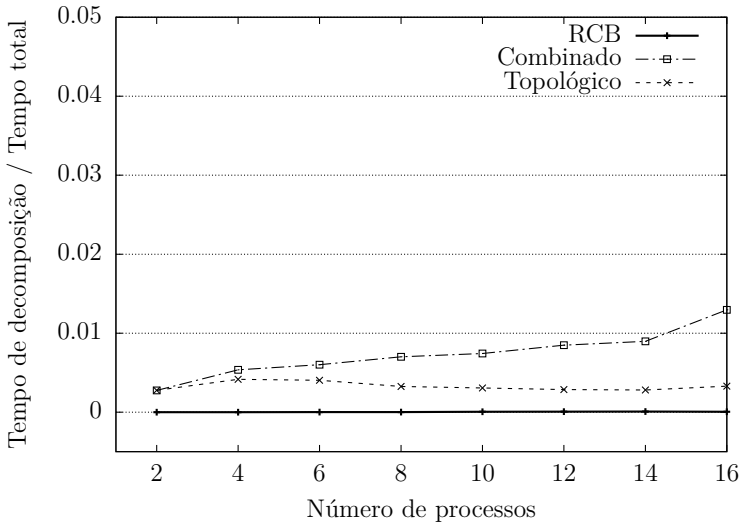
Figura 92 – Estimativa do volume de comunicação para corrida com 272.573 partículas.



Verifica-se que o particionamento baseado em *RCB* obteve os menores volumes de comunicação dentre todos os casos avaliados. O particionamento combinado obteve ainda um menor volume de comunicação que a técnica puramente topológica.

Os tempos de decomposição de domínio também foram avaliados para verificar a influência desta tarefa no tempo de processamento total, conforme apresentado na Figura 93.

Figura 93 – Tempo de decomposição para corrida com 272.573 partículas.



Observa-se que o tempo demandado nestas tarefas é muito pouco expressivo frente ao tempo de processamento total (menor que 1,5%). Além disso poucas tarefas de decomposição foram necessárias para o intervalo de tempo simulado. A estratégia topológica, por exemplo, exigiu apenas 17 particionamentos durante os 25.000 passos de integração.

8.2.3 Influência do tamanho das células de decomposição

Conforme apresentado, as técnicas de paralelização baseadas em topologia baseiam-se na definição de células auxiliares para seu devido funcionamento. No caso da técnica topológica, estas células correspondem aos subdomínios espaciais associados aos elementos das matrizes de presença de partículas nos processos. Na técnica combinada, as células são artifícios auxiliares usados para compartilhamento de partículas entre processos. Em ambos os casos, é necessária a definição das dimensões destas células regulares.

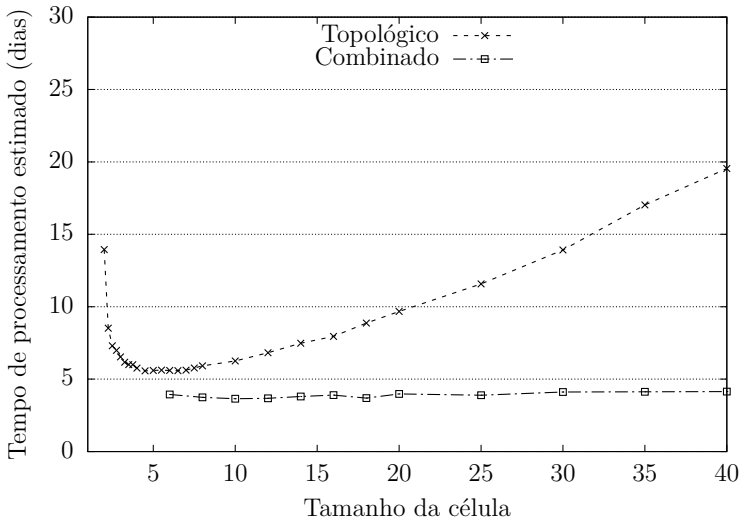
O tamanho mínimo das células deve atender a alguns requisitos mínimos para o funcionamento adequado das técnicas de paralelização. No caso topológico a dimensão mínima da célula corresponde ao diâmetro da maior partícula do modelo. Já no caso combinado, esta dimensão deve ser no mínimo três vezes o diâmetro da maior partícula.

Uma vez que a dimensão das células tem influência no volume de comunicação entre processos, é necessário quantificar esta relação. Para tanto adota-se como métrica o tempo de processamento estimado e o número médio de partículas compartilhadas entre processos.

As Figuras 94 e 95 avaliam, respectivamente, o tempo de processamento e o número médio de partículas compartilhadas como função do tamanho das células. Os dados apresentados para as dimensões de comprimento são fatores que multiplicam o

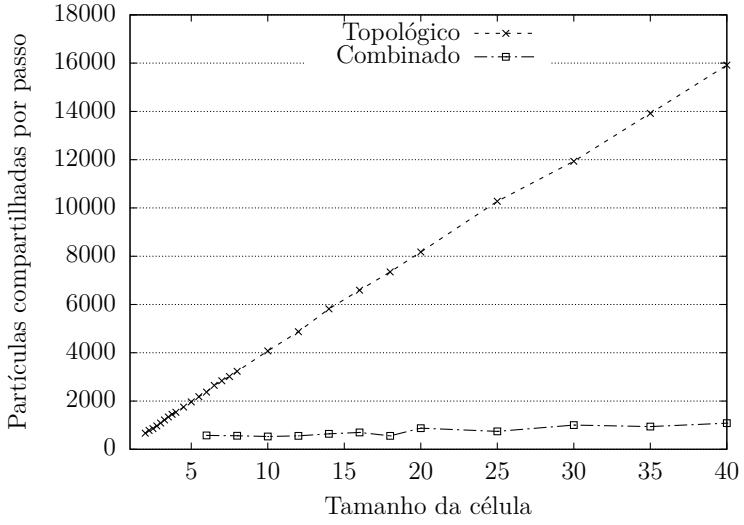
raio máximo das partículas. Logo, os valores mínimos assumidos para as técnicas de paralelização são 2, no caso topológico, e 6 no caso combinado.

Figura 94 – Influência do tamanho das células de decomposição no tempo de processamento.



Note que o tempo de processamento é bastante sensível ao tamanho da célula no caso topológico. Já no caso combinado, houve pouca relação entre eficiência e o tamanho da célula. A correlação com esta dimensão também é bastante notável em termos do número médio de partículas compartilhadas entre processo.

Figura 95 – Influência do tamanho das células de decomposição no volume de comunicação.



Observe ainda, na Figura 95, que a técnica topológica compartilha aproximadamente a mesma quantidade de partículas que a técnica combinada para células pequenas. Apesar disto, a eficiência da técnica topológica apresenta-se inferior, conforme verificado na Figura 94. A perda de eficiência pode, portanto, ser explicada pela necessidade de

comunicações globais entre processos, necessárias para o compartilhamento das matrizes de presença na técnica topológica.

9 Paralelização distribuída - RCB

9.1 Simulação de deslizamento de terra

O objetivo desta seção é apresentar resultados numéricos e análises de desempenho obtidos através do uso da técnica de paralelização distribuída com particionamento de subdomínios por RCB. O cenário de simulação é o deslizamento de terra ocorrido em Hong Kong, no dia 13 de Agosto de 1995. O evento ocorreu nas imediações das ruas Shum Wan e Nam Long Shan e envolveu um volume expressivo de sedimentos, iniciando após uma tempestade forte e prolongada. O evento resultou na destruição de construções próximas ao porto Aberdeen que foram empurradas para o mar conforme ilustra a foto do evento apresentada na Figura 96.

Figura 96 – Deslizamento de terra da rua Shum Wan, Hong Kong (1995).

Fonte: Knill (1996).

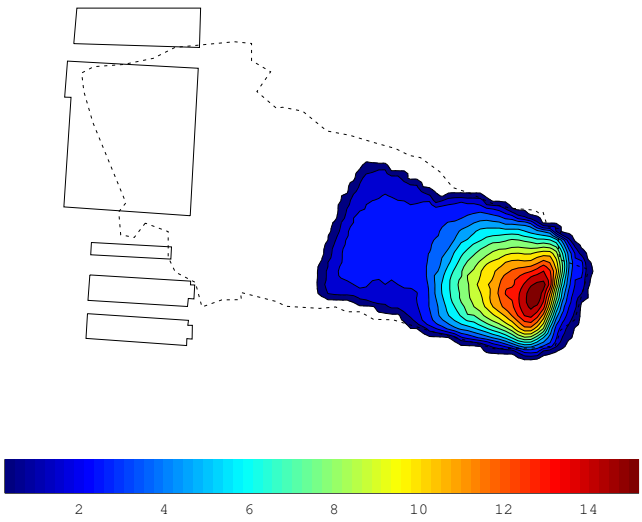
Uma investigação conduzida pelo departamento de engenharia geotécnica de Hong Kong relatou diversos detalhes relacionados com o deslizamento de terra (KNILL, 1996). Esse estudo coletou aspectos tais como condições geológicas, registros históricos de precipitação de chuva, propriedades dos materiais, superfície do terreno e deposição de detritos após a corrida de terra. Análises numéricas foram anteriormente realizadas para o deslizamento. Uma das metodologias utiliza um modelo geotécnico contínuo para representar o material durante do processo de deslizamento (WANG; SASSA, 2010). Wang e Sassa descrevem o evento como uma falha de natureza de movimento rápido que é típica de

deslizamento raso que se desloca por grandes distâncias. O coeficiente de atrito aparente foi apontado como o parâmetro chave para a representação do movimento de corrida.

Este trabalho reproduz o deslizamento de terra ocorrido em Hong Kong utilizando a formulação discreta do DEM. Um modelo tridimensional composto por partículas rígidas e obstáculos compõem o cenário de deslizamento. A massa móvel é modelada através de um conjunto de partículas esféricas, enquanto os obstáculos representam a superfície de deslizamento, a rua e algumas construções afetadas.

O modelo simulado foi construído a partir das curvas de nível do terreno fornecidas no relatório de investigação de referência (KNILL, 1996). Os dados de elevação do terreno, apresentado na planta de situação antes do deslizamento, foram obtidos desse documento. O relatório geotécnico estimou a superfície de deslizamento da corrida de detritos a partir de uma série de estudos de campo, incluindo estudos sísmicos de refração e mapeamentos geológicos. De acordo com o relatório, duas regiões definem a forma da cicatriz do deslizamento: uma plana e uma côncava, conforme ilustrado na Figura 97. A cicatriz plana está localizada na porção esquerda da região destacada. Esta região não apresenta variações significativas no perfil de profundidade que quantifica o volume de massa móvel. O mesmo não ocorre para a região côncava localizada na porção mais à direita.

Figura 97 – Perfil de profundidade da massa móvel (em metros).



Um conjunto de partículas de granulometria uniforme que discretiza a massa móvel foi criado usando um procedimento de geração geométrico. Este procedimento é baseado no arranjo BCC e permite criar a lista de partículas que compõe o domínio. A escolha deste algoritmo de geração foi baseada no tempo de processamento demandado para a criação do modelo e porque o mesmo não requer que as partículas sejam todas alocadas em memória

durante a etapa de pré-processamento Estas características são de fundamental importância para a geração dos modelos de maior escala, onde, tanto memória, quanto processamento podem ser fatores limitantes à viabilidade da geração do conjunto de partículas. A Tabela 4 apresenta o número total de partículas geradas de acordo com o nível de discretização. Note que o número de partículas cresce rapidamente com a diminuição do raio. Observe ainda que, quando se divide o tamanho da partícula por um fator de dois, o número total de partículas aumenta por um fator de aproximadamente 10.

Tabela 4 – Número de partículas de acordo com a discretização.

Raio médio	Número de partículas
$0,750$ m	4.280
$0,500$ m	17.589
$0,300$ m	96.903
$0,250$ m	176.439
$0,200$ m	363.199
$0,150$ m	904.586
$0,125$ m	1.605.530
$0,100$ m	3.220.508
$0,075$ m	7.829.115
$0,050$ m	27.097.076

Os parâmetros numéricos adotados na modelagem estão apresentados na Tabela 5. Valores diferentes de atrito de rolamento foram utilizados dependendo da região em que o contato entre partícula e obstáculo acontece.

Tabela 5 – Parâmetros de modelagem.

Parâmetro	Símbolo	Valor
Densidade das partículas	ρ_p	2,700 kg/m^3
Rigidez de contato		
Direção normal	k_n	85,000 kN/m
Direção tangential	k_t	42,500 kN/m
Fator de amortecimento crítico		
Direção normal	ζ_n	0.8
Direção tangencial	ζ_t	0.8
Atrito		
Partícula-partícula	μ	0.1
Coeficiente de atrito de rolamento		
Partícula-superfície de deslizamento	μ_r^{ps}	0.15
Partícula-superfície da rua	μ_r^{pr}	0.4
Partícula-construção afetada	μ_r^{pb}	0.5
Gravidade	g	9.81 m/s^2

9.1.1 Performance paralela

Todas as simulações numéricas foram realizadas em um *cluster* de computadores. A infraestrutura alocada possui 12 nós, cada um contendo dois processadores Intel Xeon X5570 de quatro núcleos operando a 2.93GHz e com memória cache de 8.192 KB. A quantidade de RAM disponível por nó é de 24GB, compartilhada entre 8 núcleos de processamento. Uma rede *ethernet* de 10 gigabits conecta os nós. As máquinas executam o sistema operacional CentOS 5.4 (64 bits, kernel 2.6.18), e todos os programas foram compilados usando o GCC 4.1.2 e OpenMPI 1.4.2.

Dois cenários de carga foram testados visando avaliar a eficiência paralela, *speedup* e escalabilidade. No primeiro cenário, um modelo de tamanho fixo contendo 363.199 partículas foi adotado. Este tamanho de problema pode ser resolvido serialmente em um único nó do *cluster*, então o *speedup* paralelo pode ser definido como:

$$S(m) = \frac{tp(1)}{tp(m)}, \tag{9.1}$$

e a correspondente eficiência

$$E(m) = \frac{S(m)}{m}, \tag{9.2}$$

onde $tp(1)$ é o tempo de processamento serial, e $tp(m)$ o tempo de processamento paralelo para m processos. Esta escolha de tamanho de modelo foi principalmente motivada pela limitação de memória de cada nó de processamento e o tempo total de processamento demandado na análise completa do modelo. Além disso, a eficiência paralela não pode ser definida para modelos maiores uma vez que o processamento serial não seria possível para o *hardware* empregado.

Nos testes de medição de performance, um número fixo de 22.548 passos de tempo (o equivalente a 1 s de simulação) foi adotado. A Figura 98 apresenta a eficiência obtida e a taxa de solução para cada configuração de solução paralela. A taxa de solução é uma métrica alternativa calculada a partir da divisão do tempo gasto em solução (desconsiderando comunicações) pelo tempo total de processamento. Seu valor é útil na avaliação da eficiência do programa paralelo quando o tempo de processamento serial não é conhecido ou aplicável.

Note, na Figura 98, que a eficiência paralela está abaixo de 50% quando o número de processos é maior que 16. Isto indica que o tempo de processamento está relativamente baixo quando comparado ao tempo de comunicação. O que ocorre é que o número médio de partículas por processo nestes casos é pequeno, abaixo de 12.000 partículas, como mostrado na Tabela 6, então a baixa eficiência era esperada. A Figura 98 também mostra a similaridade de comportamento para as métricas de eficiência paralela e taxa de solução.

A Tabela 6 também apresenta o tempo de processamento demandado por passo de análise, medido em segundos, e o valor de tempo teórico de referência $tp_{theor}(m)$ esperado caso a eficiência da paralelização fosse plena. Note que, quando as eficiências estão abaixo

Figura 98 – Eficiência paralela para 363k partículas.

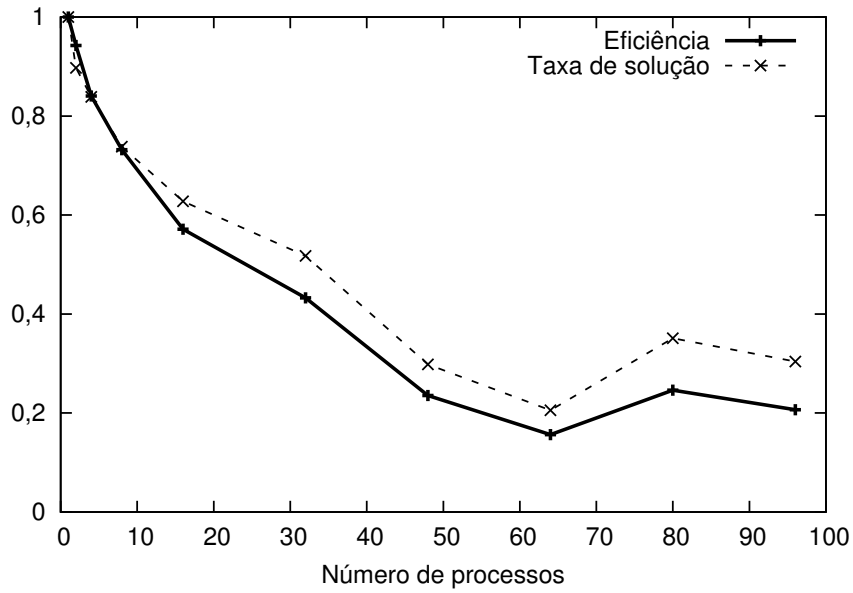


Tabela 6 – Número esperado de partículas e tempo de processamento por passo.

m	N/m	$tp(m)$/passo	$tp_{theor}(m)$/passo
1	363.199	$3,10831$ s	-
2	181.599	$1,64925$ s	$1,55416$ s
4	90.780	$0,924759$ s	$0,77707$ s
8	45.340	$0,530838$ s	$0,38854$ s
16	22.670	$0,339974$ s	$0,19427$ s
32	11.350	$0,224446$ s	$0,09713$ s
48	7.567	$0,275028$ s	$0,06475$ s
64	5.674	$0,31095$ s	$0,04857$ s
80	4.540	$0,157959$ s	$0,03885$ s
96	3.783	$0,156796$ s	$0,03238$ s

de 50%, as reduções em tempo de processamento não são relevantes. O *speedup* paralelo obtido é apresentado na Figura 99. Os dados apresentados são comparados com a referência teórica que assume que o código é integralmente paralelizado e os tempos de comunicação entre processos são desprezados.

No segundo cenário, a forma como a simulação paralela comporta-se com o aumento do número de partículas é investigada. Neste caso, o número de processos empregados é mantido constante e o tempo de processamento é calculado para três discretizações espaciais distintas em número de partículas. A Figura 100 mostra o aumento de tempo de processamento por passo de análise com o aumento do tamanho do problema.

Verifica-se um comportamento quase linear com o número de partículas. Isto acontece uma vez que o número de partículas em cada processo também aumenta com o tamanho do problema e, além disso, existe uma relação linear entre o número de partículas

Figura 99 – *Speedup* paralelo medido na simulação de 363k partículas.

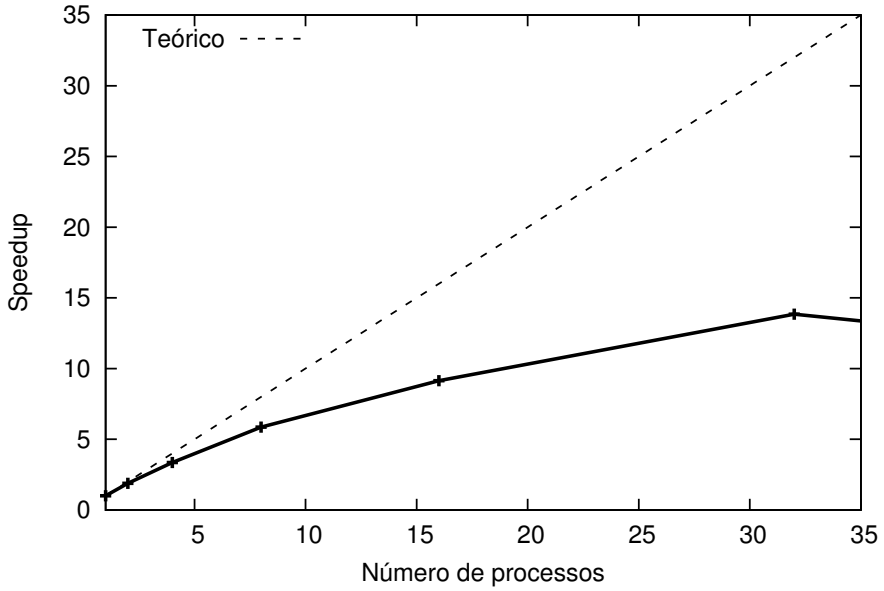
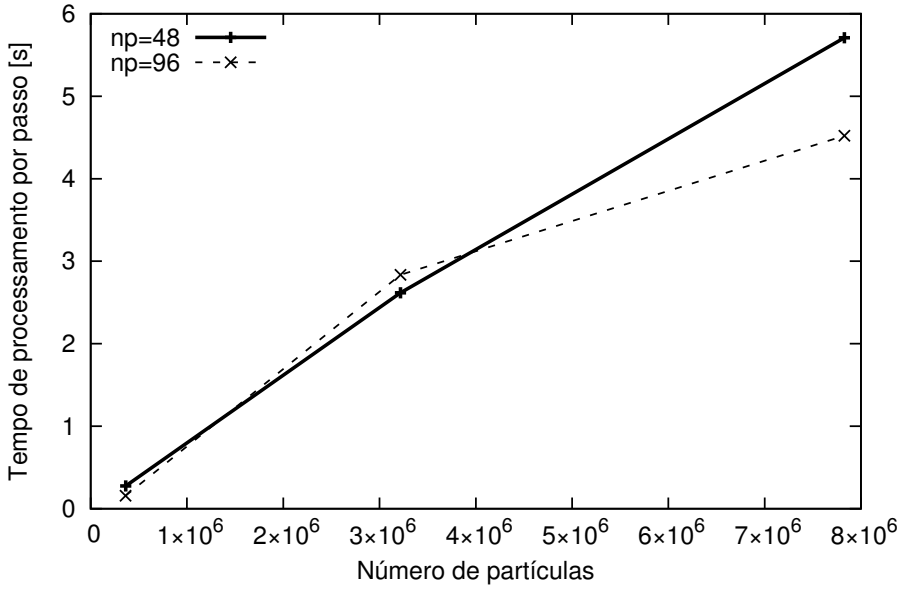


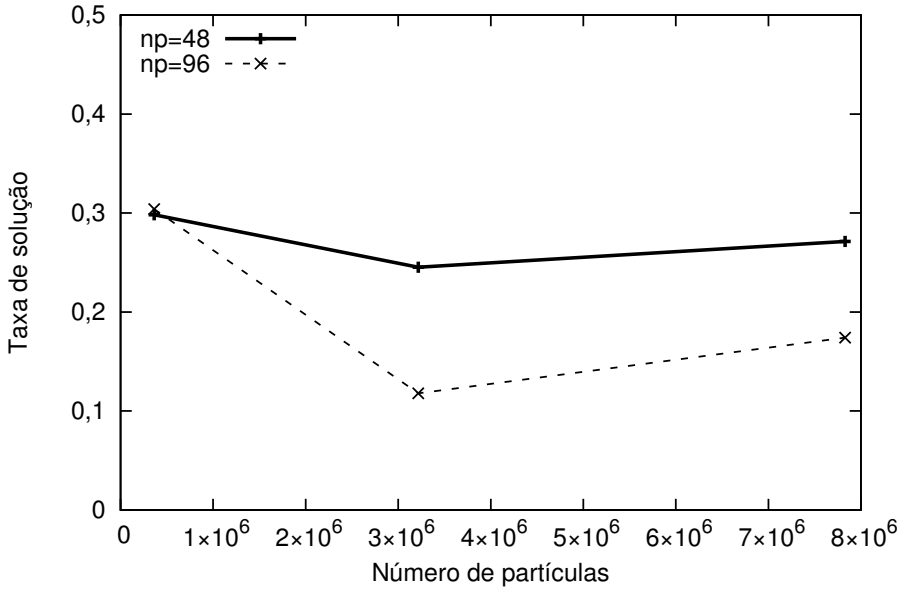
Figura 100 – Tempo de processamento por passo para o segundo cenário de testes.



e o tempo demandado de processamento. Também observa-se que o aumento no número de processos apenas resulta na redução do tempo de processamento para o maior modelo (7,8 milhões de partículas). A taxa de solução foi utilizada para avaliar a performance destes modelos de maior escala. A Figura 101 mostra como o sistema comporta-se de acordo com o tamanho do problema.

Estes resultados sugerem que a eficiência paralela está pode volta de 30% para a configuração com 48 processos, diminuindo para por volta de 15% no caso onde 96 processos foram adotados.

Figura 101 – Taxa de solução como função do tamanho do problema.



9.1.2 Balanço dinâmico de carga e comunicação

Nesta seção, o histórico de balanço de carga é apresentado para mostrar como o particionamento baseado em RCB comporta-se durante a simulação de deslizamento de terra. O teste adotado executa uma simulação paralela incluindo 16 processos durante 20 s de análise, conforme ilustrada na Figura 102. Este intervalo de tempo contém o comportamento dinâmico do problema de forma integral, desde o início do fluxo de partículas até a deposição final destes elementos.

O cálculo do desbalanço de carga considera ou não a presença de partículas vizinhas, assim como definido na Equação (3.4) e na Equação (3.5). Estas métricas permitem avaliar, tanto a qualidade do particionamento, como os volumes de comunicação envolvidos. A Figura 103 apresenta os histórico de desbalanços para o modelo com 363.199 partículas.

Os resultados apresentados mostram que o desbalanço de particionamento apresenta valores quase constantes durante a simulação. A tolerância para desbalanço $\delta_{tol} = 0,05$ é de fato respeitada. Isto indica uma boa qualidade de particionamento e um mecanismo efetivo de balanço dinâmico de carga. Verifica-se no entanto que os volumes de comunicação variam significativamente. Em certos instantes, o volume de partículas provenientes de processos vizinhos supera o existente em um dado processo ($\delta_{imbal} > 1$). No estágio final da simulação, o volume de comunicação atinge baixos patamares devido à deposição de partículas e a consequente redução na frequência de reparticionamentos de domínio.

Figura 102 – Particionamento de domínio para 16 processos.

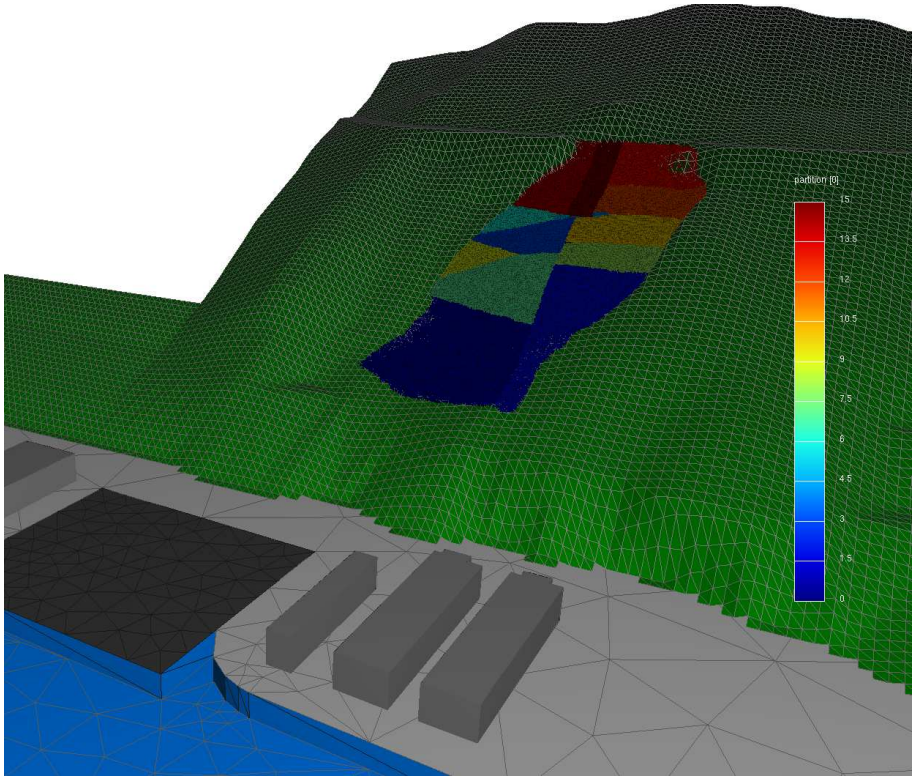
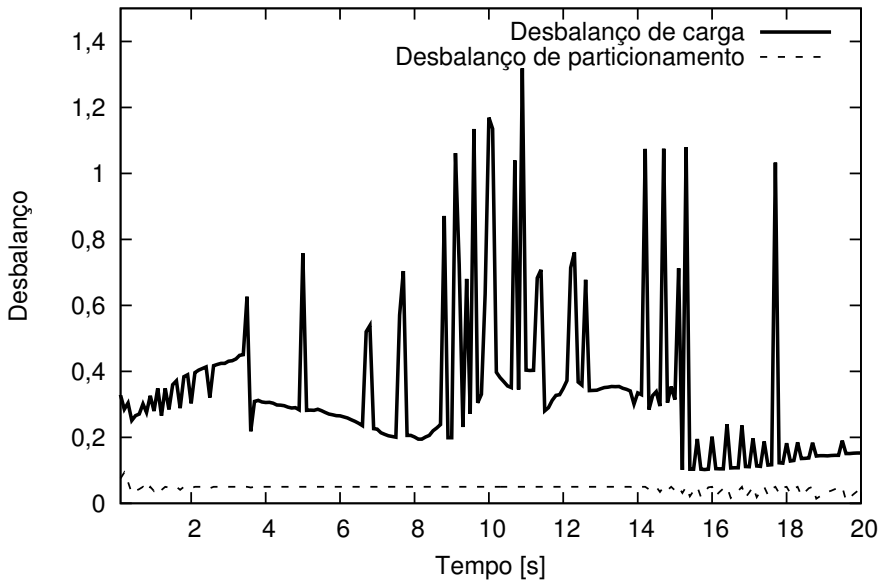


Figura 103 – Desbalanços para a simulação de 363k partículas.



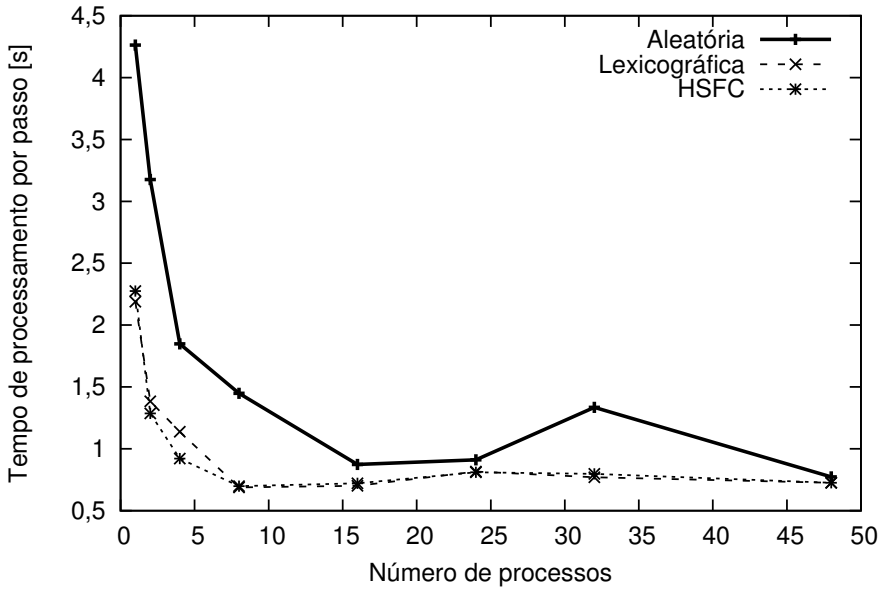
9.1.3 Otimização de acesso à memória

Os resultados até então apresentados mostram a performance das simulações paralelas considerando todas as tarefas envolvidas na metodologia apresentada na subseção 4.3.2. Visando avaliar o efeito isolado do procedimento de otimização de acesso à memória, um teste de medição de performance é analisado para o problema de deslizamento de terra. O

cenário de medição também considera a simulação do modelo contendo 363.199 partículas durante um número prescrito de 22.548 passos de tempo. Métodos distintos de controle de memória são investigados: o baseado em sequências lexicográficas e o método proposto baseado em HSFC. Os resultados são comparados com um caso de referência em que o acesso à memória é forçadamente aleatório.

A Figura 104 apresenta o tempo de processamento por passo de análise como função do número de processos paralelos e do método adotado. As medições de tempo realizadas incluem o tempo gasto pela tarefa de otimização de acesso à memória através da organização do conjunto de partículas em memória (RAM). Este estudo permite avaliar como o problema de *cache miss* depende do tamanho do modelo e a influência dos algoritmos de otimização de acesso.

Figura 104 – Tempo de processamento por passo de acordo com a ordenação de partículas.



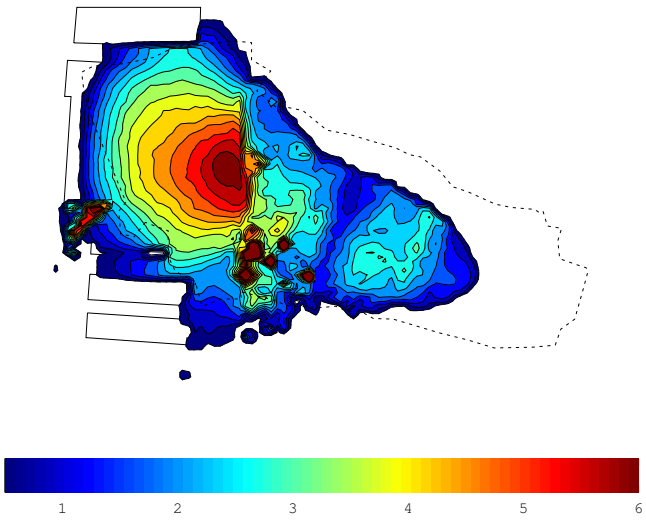
Note que o método de acesso aleatório apresenta a pior performance, com diferenças significativas em tempo de processamento. As eficiências dos métodos lexicográfico e baseado em HSFC são melhores quando o número de partículas por processo é maior, sendo o método baseado em HSFC ligeiramente mais eficiente que o método lexicográfico. Em alguns casos a diferença relativa entre estes dois métodos e o método aleatório é maior que 140%. Esta diminuição de eficiência é esperada porque a quantidade de memória alocada diminui com o aumento do número de processos. Assim sendo, os subdomínios gerados passam a caber na memória *cache* e o método de controle de memória passa a não influenciar tanto no tempo de processamento.

9.1.4 Fluxo de massa e deposição de partículas

Visando avaliar a qualidade dos resultados da metodologia de solução paralela proposta, os resultados computacionais obtidos foram analisados de acordo com parâmetros geométricos e cinemáticos para o fluxo de massa e área afetada. Os estudos comparativos utilizam a cicatriz do deslizamento de terra e os perfis de deposição em alguns pontos de medição. Adicionalmente, a cinemática da massa móvel foi investigada. As referências de comparação adotadas foram as providas pelo relatório forense elaborado depois do deslizamento de terra ocorrido (KNILL, 1996).

A Figura 105 apresenta o mapa de cores que reflete a deposição das partículas para a simulação numérica. Este mapa é construído com base em um conjunto de células de medição que divide o plano de deposição das partículas. Considerando que uma partícula seja unicamente projetada em uma célula de medição de acordo com sua posição, a altura de deposição é definida como a diferença entre a máxima e a mínima elevação das partículas de cada célula.

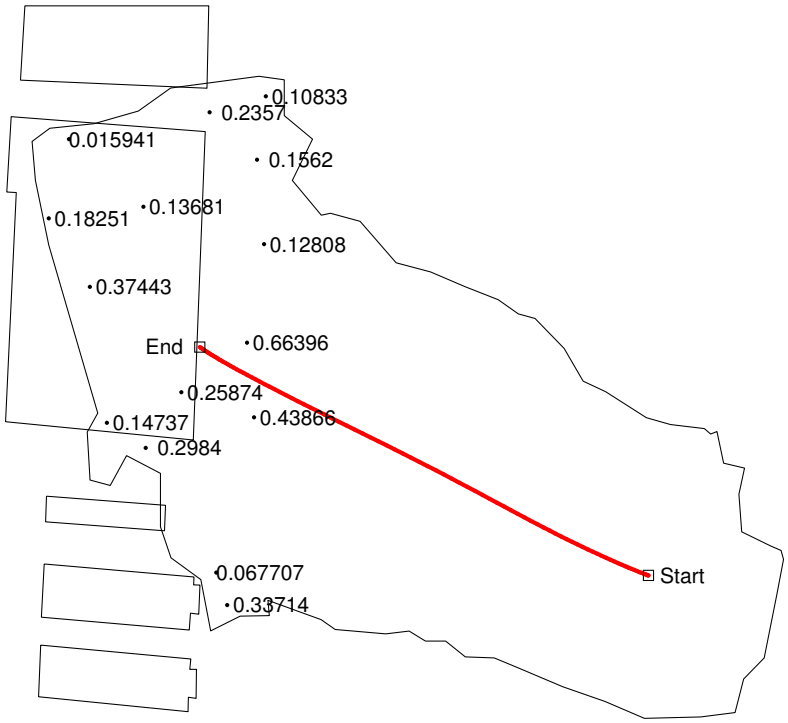
Figura 105 – Mapa de cores para as alturas de deposição da simulação numérica (medidas em metros).



Note que grande parte do volume de deposição está localizado dentro das fronteiras que delimitam a construção afetada pelo escorregamento de terra. Além disso, a massa móvel está praticamente depositada dentro da cicatriz do deslizamento relatada para o evento real ocorrido em Shum Wan (região pontilhada). Em alguns locais, a altura de deposição de terra foi da ordem de seis metros. Considerando todas as simplificações de modelagem utilizadas neste trabalho, obteve-se uma boa aproximação para o padrão de deposição através do modelo numérico baseado no DEM. Um comparativo numérico para

o perfil de deposição de terra é também apresentado na Figura 106. As diferenças relativas para as deposições obtidas no caso real e no modelo numérico são apresentadas em alguns pontos de medição. Estes valores oscilam entre 1.26% e 66.4%, sendo 23.7% o valor médio.

Figura 106 – Diferenças relativas para as alturas de deposição / Trajetória do centro de massa.



A Figura 106 também mostra a trajetória do centro de massa em projeção. Em geral, trata-se de um movimento com caminho aproximadamente retilíneo que iniciou em cima do morro e terminou próximo à construção afetada. Os deslocamentos obtidos desde sua posição inicial são reportados, para os eixos coordenados, na Figura 107. Apresentam-se ainda as componentes de deslocamento em ambas direções longitudinal e vertical, representadas respectivamente como L e H. Estas direções são comumente utilizadas em análises de deslizamento de terra.

De acordo com a Figura 107, os deslocamentos na direção longitudinal são dominantes. No estágio final da simulação, a razão H/L é igual a $0,402$. As velocidades para o deslizamento são apresentadas na Figura 108 para o centro de massa.

A velocidade apresentada também é dominante para a direção longitudinal. Aproximadamente no instante $t = 9.4$ s, a magnitude da velocidade atinge seu valor máximo $v_{\max} = 12.57$ m/s. O conjunto de partículas neste instante é apresentado na Figura 109. Estes resultados são comparáveis em termos de componentes de velocidade aos apresentados por Wang e Sassa (2010) em seu modelo geotécnico contínuo.

Figura 107 – Deslocamento do centro de massa.

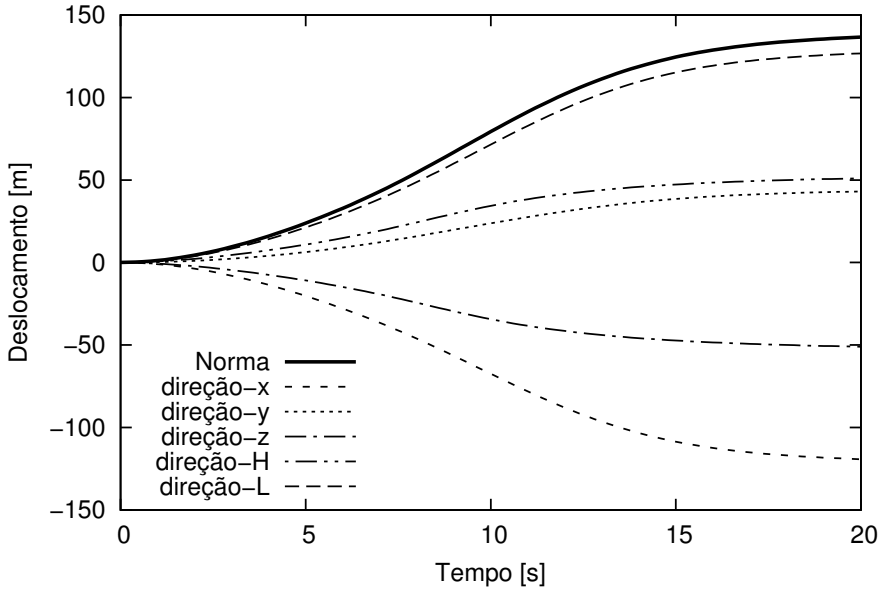
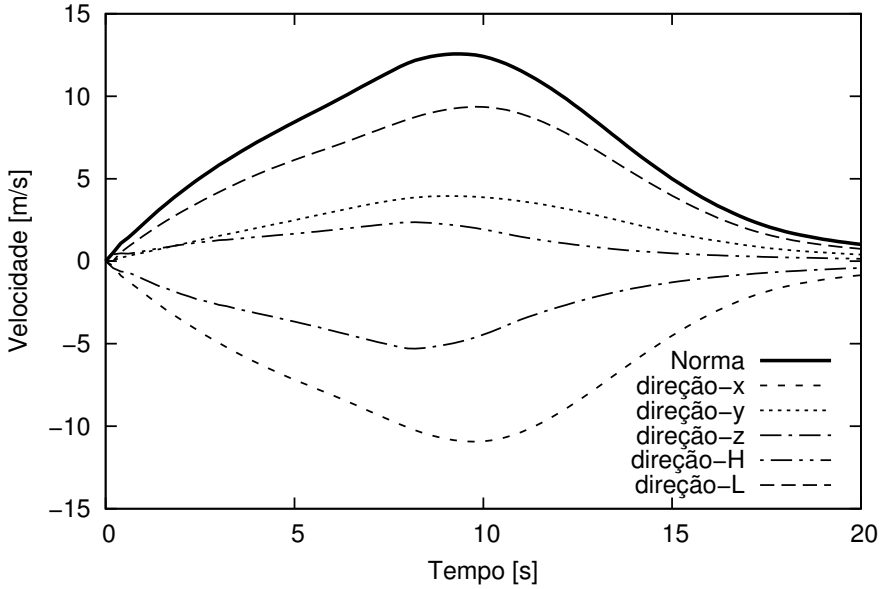


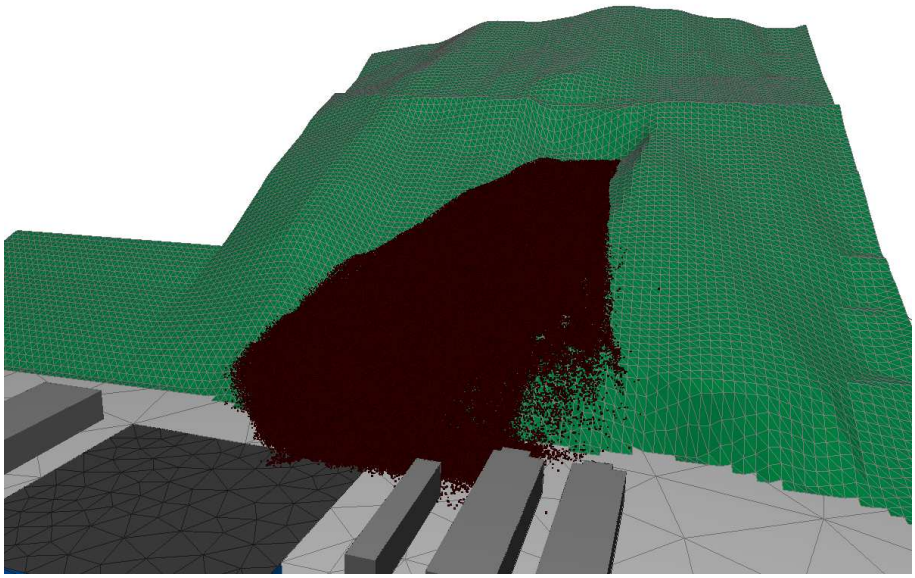
Figura 108 – Velocidade do centro de massa.



9.2 Considerações

Os resultados numéricos apresentados neste capítulo foram obtidos para uma simulação computacional que tenta reproduzir um problema prático de engenharia. O cenário de aplicação adotado apresenta diversas dificuldades associadas tanto com a modelagem do domínio quanto com a simulação dinâmica propriamente dita. Esta aplicação lida com um domínio computacional altamente espalhado e com movimentos rápidos para as partículas que representam o meio. Estas características demandam fortemente o emprego combinado das técnicas de balanço dinâmico de carga, otimizações relacionadas com a busca de contatos e a questão de localidade dos dados em memória. Sem o uso

Figura 109 – Configuração do modelo de partículas no instante $t = 9.4$ s.



destes mecanismos, integrados à técnica de paralelização distribuída, a simulação não seria possível em tempo hábil.

Verificou-se que as técnicas de controle de acesso à memória aceleram consideravelmente o tempo de processamento. A melhoria de performance obtida pelo melhor uso da hierarquia de memórias *cache* tende a aumentar com o tamanho do problema, mais especificamente com o aumento do número de partículas por processo paralelo.

O método de particionamento baseado em RCB conferiu à simulação partições de qualidade, resultando em desbalanços de particionamento abaixo de 5% durante toda a simulação. Isso reflete a boa compatibilidade do método com a estratégia de controle de balanço de particionamento adotada.

Os estudos de escalabilidade demonstram que a eficiência paralela tende a aumentar com a elevação do número de partículas por processo, ficando acima de 70% para simulações contendo acima de 45.000 elementos por processo. Além disso, a relação linear entre o número de partículas e o tempo de processamento mantém-se quando o número de processos é mantido constante. Isto indica a boa eficiência do algoritmo de busca de contatos, mesmo para este cenário onde o domínio computacional é bastante espalhado.

Por fim, os resultados da simulação de deslizamento demonstram uma boa aproximação do modelo numérico à realidade, mesmo considerando todas as simplificações de modelagem e a natureza do problema. Os comparativos numéricos conduzidos mostram uma diferença relativa entre a simulação numérica e as medições de campo abaixo de 25%, considerando valores médios.

10 Paralelização híbrida

Visando investigar a performance paralela das estratégias de solução baseadas em modelos híbridos de decomposição do domínio computacional, quatro estudos distintos são analisados.

- *Fluxo de partículas*: Este exemplo reproduz um fluxo de partículas em uma rampa inclinada tal como apresentado na Figura 110 (a). Inicialmente, um conjunto de partículas está em repouso no topo da região inclinada da rampa e contido por uma parede móvel perpendicular à superfície da rampa. Repentinamente, esta parede é removida e as partículas começam então a movimentar-se devido a ação de forças gravitacionais. O movimento finaliza com a deposição destes elementos discretos no plano horizontal. Este experimento numérico auxilia na validação numérica do modelos do DEM uma vez que tanto o experimento como a simulação numérica podem ser desenvolvidas. No contexto deste trabalho, o experimento numérico foi escolhido com o intuito de servir como referência para o estudo de *profile* das rotinas computacionais.

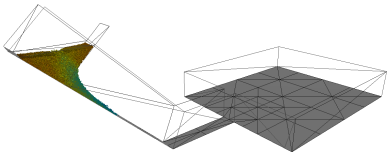
- *Experimento Box-in-a-box*: O segundo caso de teste foi apresentado inicialmente por Berger et al. (2015) com o intuito de avaliar questões relacionadas com o balanço de carga em decomposições de subdomínios. Este teste utiliza dois cubos concêntricos definidos por paredes planas e contidas em planos paralelos aos planos cartesianos. O cubo interno tem 30 cm de largura de aresta enquanto que o externo possui 50 cm de aresta. Um conjunto de 50.000 partículas com diâmetro de 5 mm é inserido no topo do domínio. Devido à gravidade, estas partículas caem e atingem a face superior do cubo interno, bem como as faces laterais e a base do cubo externo, tal como apresentado na Figura 110 (b).

- *Deslizamento de terra*: Este caso de teste reproduz a modelagem numérica realizada para o deslizamento de terra ocorrido na rua Shum Wan, previamente discutido na seção 9.1. O objetivo deste teste é avaliar o desempenho das estratégias de paralelização híbrida quando comparadas com as execuções puramente distribuídas. O teste também avalia se o número de obstáculos afeta a performance paralela. A Figura 110 (c) ilustra o modelo numérico empregado.

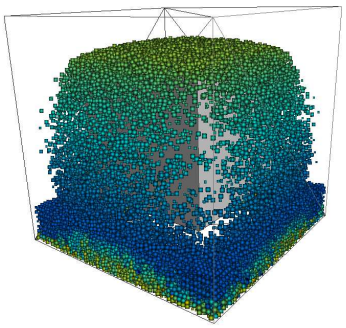
- *Funil de descarga*: O último teste de caso tenta reproduzir o experimento numérico realizado por Markauskas e Kačeniauskas (2015). Este exemplo simula o fluxo de um conjunto de partículas dentro de um funil. O estudo de escalabilidade do modelo híbrido de programação apresentado neste trabalho é o objetivo principal deste caso.

Experimentos numéricos de modelos de larga escala incluindo até 3,2 milhões de partículas foram realizados.

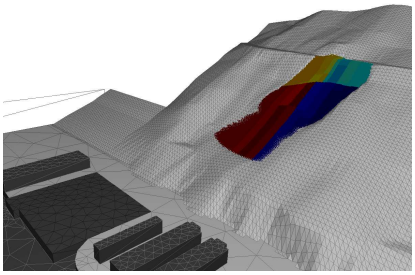
Figura 110 – Estudos de caso para avaliação das estratégias híbridas de paralelização.



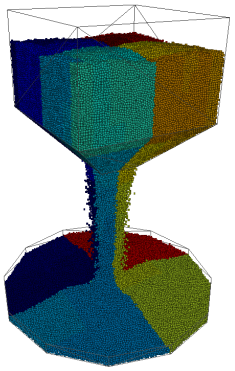
(a) Fluxo de partículas em rampa inclinada.



(b) Experimento *Box-in-a-box*.



(c) Deslizamento de terra.



(d) Funil de descarga.

Os estudos de performance paralela utilizam as métricas de *speedup*, eficiência, balanço de carga, dentre outras. A estrutura computacional alocada é a mesma detalhada previamente na subseção 9.1.1. Nos testes realizados neste capítulo, no entanto, o balanço de carga medido considera o número de testes geométricos de contato ao invés do número de partículas. Esta forma permite avaliar melhor a carga de trabalho associada às *threads* durante o procedimento de busca de contato. Trata-se de uma métrica mais apurada de avaliação uma vez que o número de testes geométricos para as partículas não é mesmo, podendo variar ao longo do tempo e da posição no domínio. Esta métrica pode ser definida para o processo i no instante de tempo t como:

$$\omega_{it} = \frac{n \times \max(gt(c))}{\sum_{c=1}^{n} gt(c)}, \tag{10.1}$$

onde $gt(c)$ é o número de testes geométricos realizados pela *thread* c e n é o número de *threads* empregadas. O desbalanço médio de carga considerando toda a sequência histórica de desbalanços para os subdomínios i pode ser definido através da métrica global:

$$WI = \frac{1}{s} \sum_{t=1}^{s} \sum_{i=1}^{m} \frac{\omega_{it}}{m} \tag{10.2}$$

Tabela 7 – Fluxo de partículas considerando 17.000 elementos, 2 processos MPI e 8 *threads* por processo.

	HSFC	RCB	METIS
Tempo de processamento	2806, 77 s	3255, 44 s	3514, 38 s
Desbalanço de carga	1,0067	1,0122	1,0979
Número médio de testes geométricos	226558	226738	227462
Número de testes geométricos nas interfaces	48326	35701	33147
Cálculo de forças internas	1781, 28 s	1750, 96 s	1764, 25 s
Cálculo de forças externas	11, 11 s	9, 12 s	10, 89 s
Integração temporal	42, 06 s	37, 31 s	40, 48 s
Definição de blocos por threads	4, 34 s	276, 33 s	288, 54 s
Controle de memória PSS	0, 00 s	18, 60 s	0, 00 s
Controle de memória HSFC	0, 24 s	0, 00 s	0, 06 s
Balanço de carga por thread	10, 10 s	0, 00 s	0, 00 s
Particionamento RCB	0, 00 s	16, 40 s	0, 00 s
Particionamento METIS	0, 00 s	0, 00 s	17, 95 s

onde s é o número de passos de tempo utilizados na medição de performance e m é o número de processos adotados na decomposição de domínio.

10.1 Corrida em rampa

O experimento de fluxo de partícula foi realizado para dois arranjos distintos de partículas contendo 17.000 e 39.000 elementos. Estas simulações apresentam alta mobilidade de partículas e permitem avaliar aspectos relevantes das estratégias de solução híbrida adotadas. Em especial, a natureza do problema exige que a solução paralela realize reparticionamentos contínuos para os subdomínios e a consequente ordenação das informações de partícula. Os cálculos numéricos utilizam 80.849 passos de tempo para representar $3,5$ s de simulação. Os relatório de *profile* são apresentados na Tabela 7 para cada estratégia de solução. Esta informação permite avaliar o quanto as tarefas principais demandadas pela solução paralela influenciam na composição do tempo de processamento total. A tabela apresenta, além de tempos de execução, outras variáveis que revelam o balanço de carga obtido. Estas informações são separadas por coluna de acordo com a estratégia de solução empregada.

Os resultados apresentados na Tabela 7 mostram que o método HSFC apresenta melhor desempenho que as outras estratégias tanto para tempo de processamento quanto para balanço de carga. Observa-se ainda o número reduzido de testes geométricos nas interfaces entre *threads*, valor abaixo de 25% do total de testes geométricos reportados por passo de tempo para este problema de pequena escala. Embora o método topológico (METIS) tenha performance melhor que os métodos geométricos considerando este aspecto, a redução do número de testes geométricos nas interfaces não esteve associado a um

Tabela 8 – Fluxo de partículas considerando 39.000 elementos, 2 processos MPI e 8 *threads* por processo.

	HSFC	RCB	METIS
Tempo de processamento	5741, 09 s	7549, 69 s	8631, 44 s
Desbalanço de carga	1,0042	1,0078	1,0755
Número médio de testes geométricos	558212	558708	560382
Número de testes geométricos nas interfaces	90783	70646	63073
Cálculo de forças internas	3712, 12 s	3741, 11 s	3840, 22 s
Cálculo de forças externas	41, 57 s	38, 65 s	47, 67 s
Integração temporal	105, 79 s	111, 30 s	132, 98 s
Definição de blocos por threads	10, 08 s	964, 24 s	1046, 31 s
Controle de memória PSS	0, 00 s	47, 12 s	0, 00 s
Controle de memória HSFC	0, 38 s	0, 00 s	0, 07 s
Balanço de carga por thread	25, 55 s	0, 00 s	0, 00 s
Particionamento RCB	0, 00 s	41, 45 s	0, 00 s
Particionamento METIS	0, 00 s	0, 00 s	47, 12 s

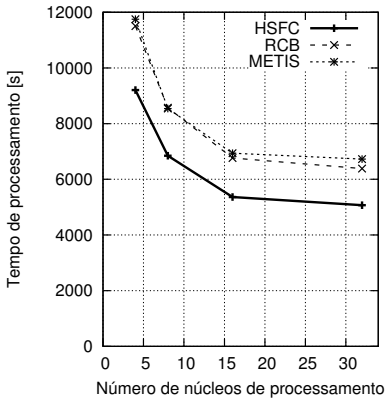
bom balanço de carga. Os tempos de execução relacionados com os cálculos de forças externas e internas assim como o procedimento de integração temporal são relativamente próximos para todos os métodos testados. A mesma similaridade de comportamento pode ser observada para a tarefa de particionamento local.

Em contraste, as tarefas de definição de *threads* diferem proporcionalmente em tempo quando se compara o método baseado em HSFC com os métodos RCB e METIS. Esta constatação indica que o método HSFC é menos afetado por reparticionamentos contínuos e operações de ordenação para o vetor de partículas. Considerando experimentos de diferentes tamanhos de modelo, a tarefa de definição de *threads* é mais custosa no experimento de maior escala, conforme sugere a Tabela 8. Entretanto, a estratégia baseada em HSFC ainda apresenta uma boa performance quando comparada com os métodos baseados em RCB e METIS. A partir da Tabela 8 é possível ainda calcular o custo associado com a viabilização da solução que utiliza HSFC (*overhead*). Neste caso, verifica-se que a metodologia demanda um tempo adicional abaixo de um porcento do tempo total de execução. A Tabela 8 repete o estudo de *profile* para um modelo maior, envolvendo 39.000 partículas.

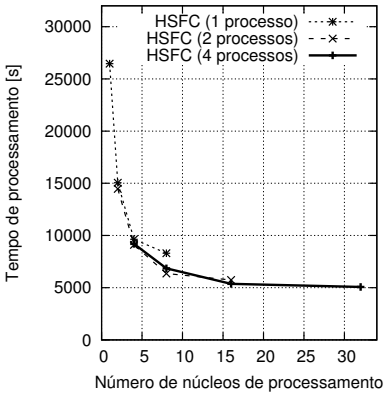
A Figura 111 apresenta um comparativo de execuções paralelas considerando o número de núcleos de processamento empregados e as estratégias de solução. Observa-se um melhor desempenho para o método HSFC nas distintas configurações de execução realizadas para o problema de maior escala.

A Figura 111 (a) mostra que os tempos de execução para o método HSFC estão abaixo daqueles obtidos pela estratégia RCB e METIS. A utilização do modelo híbrido de execução, quando processos MPI e *threads* OpenMP operam em conjunto, permite

Figura 111 – Tempo de processamento para o fluxo de 39.000 partículas.



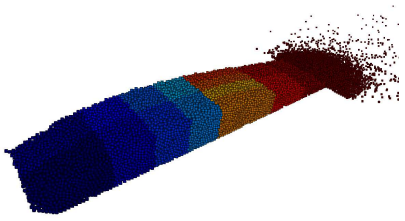
(a) Comparativo de algorítimos (4 processos).



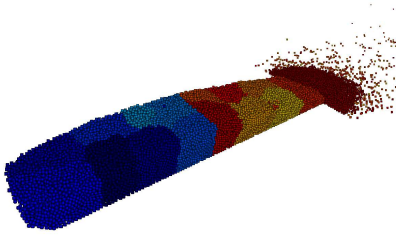
(b) HSFC.

uma melhoria da performance paralela. A Figura 111 (b) apresenta a avaliação da escalabilidade forte do experimento. Este estudo indica que as execução híbridas atingem melhor performance que aquelas baseadas apenas em memória compartilhada (1 processo com paralelização OpenMP). Observa-se ainda *speedup* limite por volta de 16 núcleos de processamento. A Figura 112 ilustra a aplicação das estratégias de particionamento local para a simulação de fluxo de partículas.

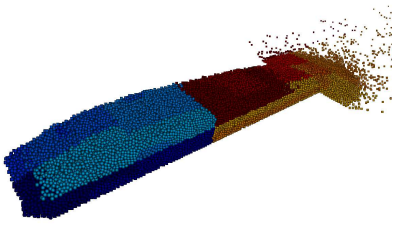
Figura 112 – Particionamento local de acordo com a estratégia de solução.



(a) RCB.



(b) METIS (Topológico).



(c) HSFC.

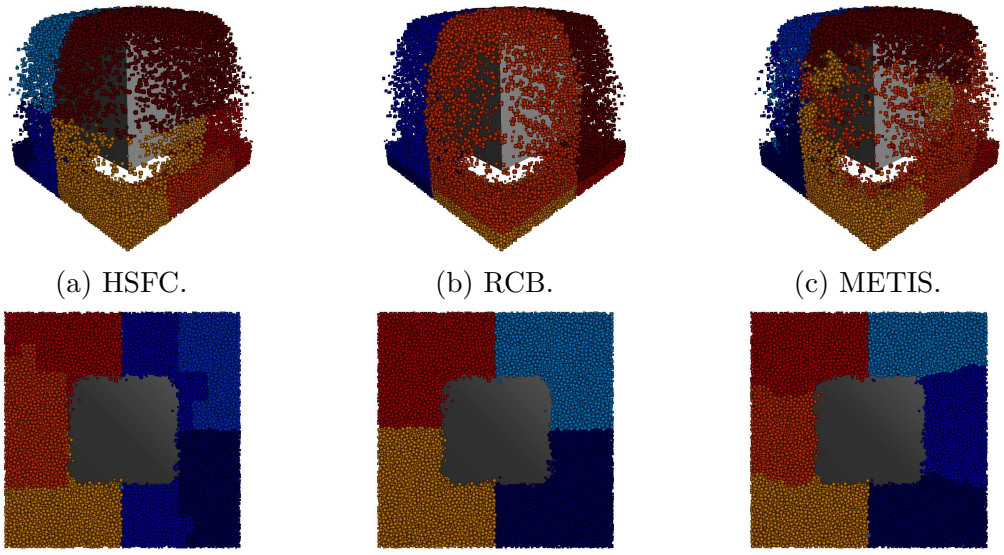
10.2 Box-in-a-box

O experimento *box-in-a-box* foi inicialmente proposto por Berger et al. (2015). Trata-se de uma simulação que retrata a problemática do balanço de carga decorrente

de decomposições de subdomínios. Isto ocorre uma vez que existe uma predominante existência de contatos em uma das regiões do domínio. Mais especificamente, a região localiza-se acima da face superior do cubo interno, conforme retrata a Figura 110 (b). Os estudos conduzidos nesta seção visam avaliar o particionamento obtido, a quantidade de vizinhos de subdomínios, desbalanços de carga e finalmente o desempenho paralelo. Os testes de medição calculam a cinemática de um conjunto de 50.000 partículas durante 50.000 passos de tempo.

A Figura 113 mostra um estágio intermediário das simulações para as estratégias de solução empregadas. As cores apresentadas na figura representam oito blocos de partículas que dividem o domínio. A decomposição empregada utiliza dois subdomínios e quatro blocos de *threads* por subdomínio. Observa-se que as estratégias de particionamento tentam agrupar as partículas em blocos utilizando alternativas distintas. O método RCB, representado na Figura 113 (b) e Figura 113 (e), apresenta em geral partições mais regulares que as estratégias baseadas em HSFC e METIS.

Figura 113 – Experimento *box-in-a-box* no instante 0.315 s.



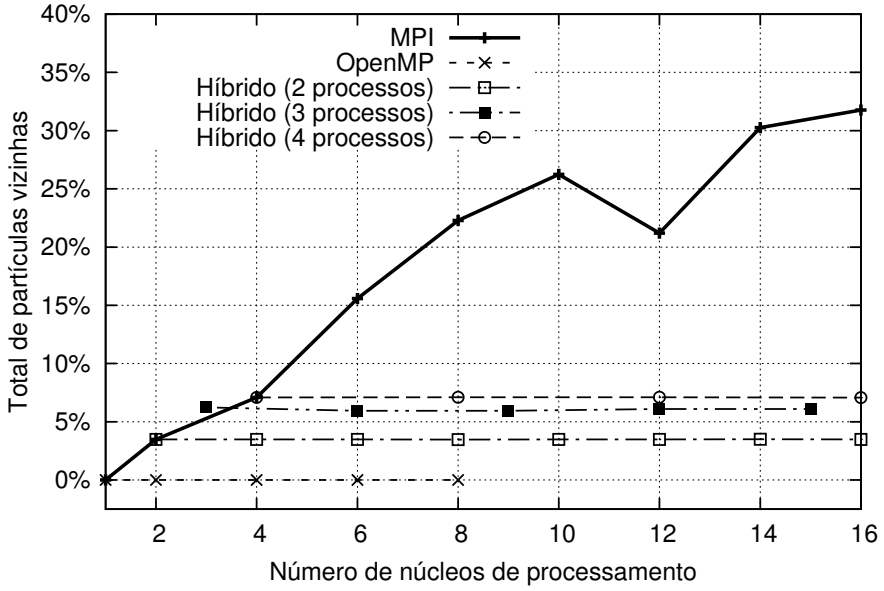
(a) HSFC. (b) RCB. (c) METIS.

(d) HSFC (vista inferior). (e) RCB (vista inferior). (f) METIS (vista inferior).

O agrupamento de partículas de um mesmo subdomínio ou *thread* é desejado nos procedimentos de particionamento locais e decomposição de domínio. Isto indiretamente tende a reduzir as interfaces entre subdomínios ou blocos de partículas e, consequentemente, melhorar a performance computacional paralela. O modelo híbrido de decomposição de domínio pode configurar diferentes modos de execução através da alteração do número de processos e *threads* responsáveis pelo cálculo dos elementos discretos. Dependendo do *hardware* disponível, é possível dar preferência ao uso de *threads* ao invés de processos. Essa prática reduz o volume de comunicação, aqui expresso pelo número de partículas "fantasmas"que tem origem em subdomínios vizinhos. A Figura 114 apresenta o número

médio de partículas fantasmas em um subdomínio considerando execuções paralelas, envolvendo até 16 processos.

Figura 114 – Número de partículas fantasmas como função do método de solução.

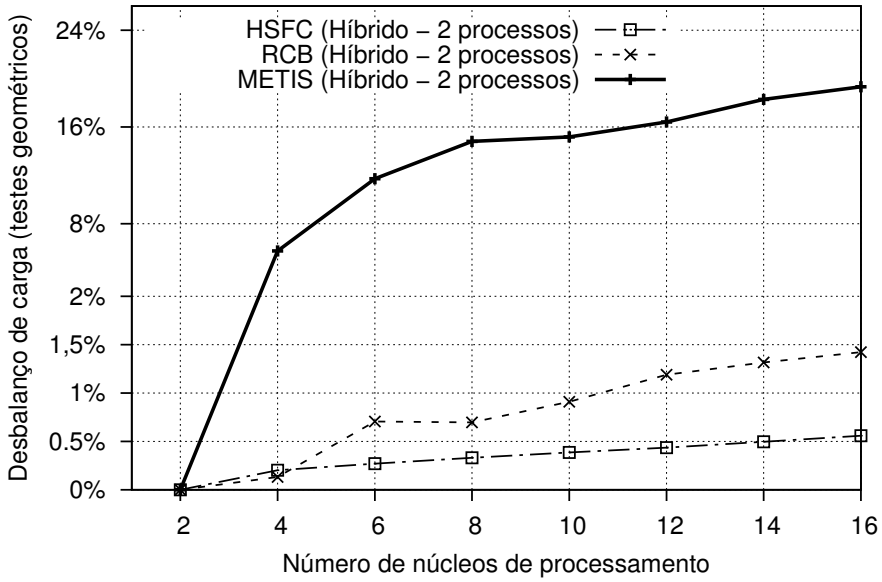


A quantidade de partículas provenientes de outros processos é inferior nas execuções híbridas quando comparadas com as simulações que não utilizam *threads*, apresentado na Figura 114 com a legenda "MPI". As execuções baseadas apenas em MPI são aquelas em que o número de *threads* por subdomínio é unitário. O modo de execução baseado apenas em *threads* é o caso especial de execução do modelo híbrido em que o número de processos é um. Este modo, apresentado na Figura 114 com a legenda "OpenMP", não possui partículas vizinhas provenientes de outros processos uma vez que, neste caso, não é necessária a decomposição de domínio. No entanto, o número de núcleos de processamento que pode ser adotado é limitado pela quantidade de processadores disponível por nó do *cluster*.

Ambas as execuções, híbrida e puramente baseada em *threads*, precisam particionar subdomínios para criar blocos de partículas com cálculos paralelos associados. Este procedimento afeta o tempo de computação por conta da estratégia de duplicação de contato discutido na seção 6.4. Os desbalanços de carga obtidos nas simulações numéricas são apresentados na Figura 115 considerando valores percentuais. O teste de medição de performance adotado utiliza apenas dois processos.

O método baseado em HSFC apresenta valores mais baixos de desbalanços para o teste realizado, indicando um bom particionamento local. A estratégia baseada em particionamento local por RCB obtém um desempenho intermediário, seguido pela estratégia topológica baseada em METIS. Ambos desbalanço de carga e volume de comunicação obtidos impactam na eficiência paralela. Os ganhos obtidos pelas execuções paralelas

Figura 115 – Desbalanço de carga baseado no número de testes geométricos.



(*speedups*) são apresentados na Figura 116.

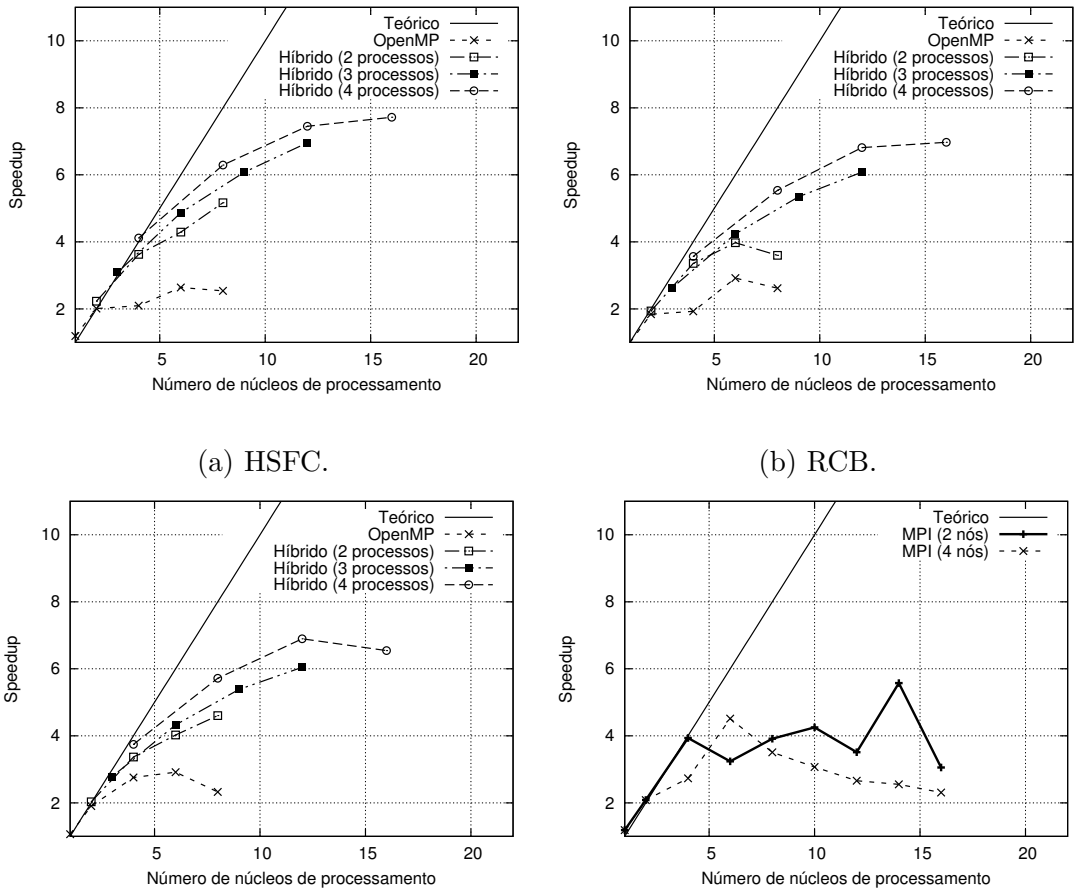
Embora estas simulações paralelas empreguem um número reduzido de partículas, as decomposições híbridas obtiveram melhores eficiências paralelas. O *speedup* medido na solução paralela baseada em HSFC foi próximo de oito para 16 processos, indicando um eficiência por volta de 50%. Esta performance é ligeiramente superior aos demais métodos testados.

10.3 Deslizamento de terra

O experimento de simulação conduzido nesta seção refere-se ao modelo de deslizamento de terra previamente apresentado na seção 9.1. O objetivo do experimento aqui conduzido é avaliar como as soluções híbridas de paralelização comportam-se para o modelo numérico. O estudo também permite avaliar como o número crescente de obstáculos afeta a performance paralela. No modelo selecionado, uma coleção de 363.199 partículas e 19.171 obstáculos triangulares representa a massa móvel e a superfície de deslizamento da corrida. Um número fixo de 22.548 passos de tempo foi utilizado nos testes de medição de eficiência paralela (1 s de simulação). O algoritmo de busca de contato empregado considera que as estruturas de dados baseadas em células de mapeamento podem demandar bastante recurso de memória para este tipo de problema. Desta forma, o procedimento de busca de contatos adotado considera o procedimento de busca descrito na subseção 2.4.3.

A Figura 117 (a) mostra que as execuções híbridas apresentam melhores eficiências que aquelas que utilizam unicamente MPI. A diferença, medida em termos de *speedup*, aumenta com o número de núcleos de processamento empregados. Considerando a utilização

Figura 116 – Escalabilidade forte do experimento *box-in-a-box*.



(a) HSFC.

(b) RCB.

(c) METIS.

(d) Somente MPI.

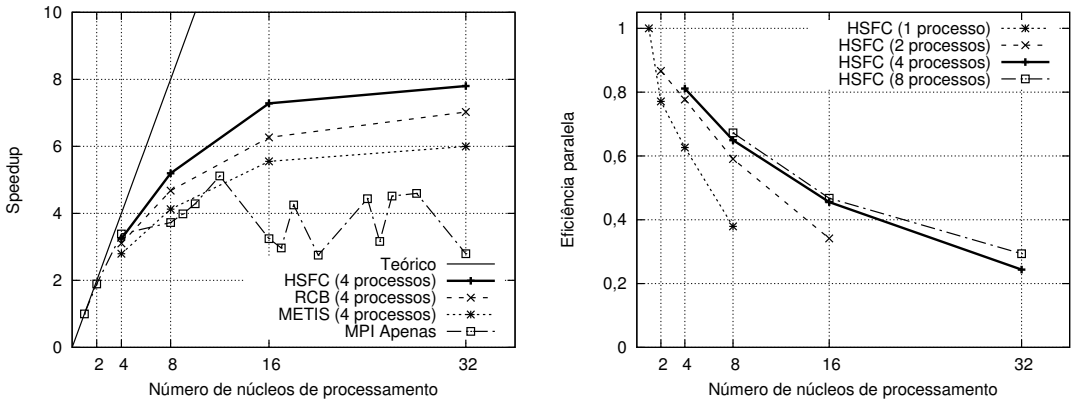
de 16 núcleos de processamento e a estratégia de solução baseada em HSFC, a eficiência paralela obtida está em torno de 50% tal como apresentado no experimento *box-in-a-box* (Figura 117 (b)). Isto indica que o aumento no número de obstáculos não afeta significativamente a performance paralela neste caso.

Observa-se ainda que as execuções paralelas que envolvem além de 16 núcleos de processamento não reduzem o tempo computacional de forma significativa, tal como apresentado na Figura 118 (a). A redução em tempo de processamento por passo de tempo é inexpressiva mesmo quando o número de processadores utilizados dobra de 16 para 32. A Figura 118 (b) sugere que, na maioria dos casos, eficiências paralelas mais altas podem ser obtidas quando o número de partículas é maior que 45.000.

10.4 Funil de descarga

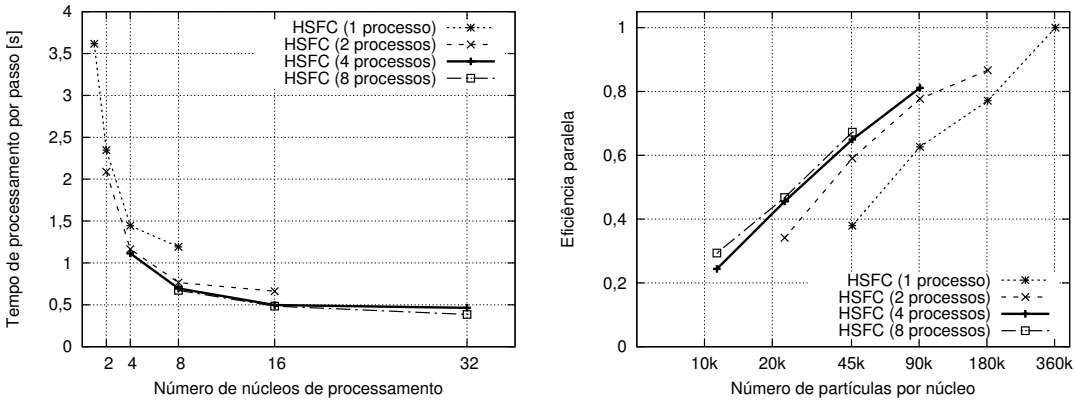
Este teste de caso visa reproduzir um modelo numérico similar ao apresentado por Markauskas e Kačeniauskas (2015). Trata-se da simulação de um fluxo de partículas em um funil que ocorre devido à abertura de um orifício, conforme ilustrado na Figura 110 (d).

Figura 117 – Deslizamento de terra - *speedup* e eficiência paralela para execuções híbridas.



(a) *Speedup*. (b) Eficiência paralela.

Figura 118 – Deslizamento de terra - escalabilidade forte para execuções híbridas.



(a) Tempo de processamento por passo. (b) Eficiência paralela.

O modelo considera um conjunto contendo 50 obstáculos triangulares e diferentes arranjos de partículas contendo desde 320.000 até 3.200.000 elementos. Os obstáculos triangulares são rígidos e imóveis e representam o funil e a superfície de deposição. O conjunto granular contém partículas esféricas não coesivas com distribuições uniformes de raios.

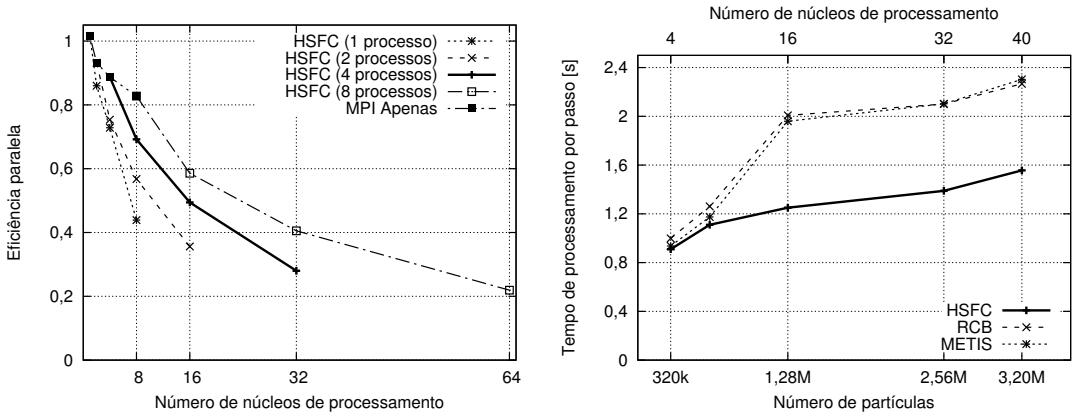
Os ensaios numéricos de performance são reproduzidos com o intuito de avaliar a escalabilidade forte e fraca do método híbrido de paralelização. Um total de 1.000 passos de tempo foi utilizado nas medições de tempo.

No primeiro experimento, o arranjo contendo 320.000 partículas é processado para diferentes configurações de execução híbrida considerando até 32 núcleos de processamento. Isto permite investigar a escalabilidade forte das implementações.

A Figura 119 (a) mostra que execuções deste caso de teste atingem altas eficiências paralelas quando o número de núcleos de processamento empregado está abaixo de oito na

maioria dos casos. Isto corresponde a um número médio de partículas em torno de 40.000 por núcleo de processamento. Constatação similar também foi observada no teste de caso que simula o deslizamento de terra (seção 10.3).

Figura 119 – Funil de descarga - Escalabilidades forte e fraca.



(a) Escalabilidade forte.

(b) Escalabilidade fraca.

Pode-se observar também que a utilização de mais processos MPI, considerando os cenários simulados, resultou em melhores eficiências. Isto pode ser explicado pela redução do número de partículas por subdomínio quando mais processos estão presentes. Embora este fato induza mais comunicações e consequentemente menos eficiência, o mesmo também acelera o procedimento de busca de contatos uma vez que o conjunto de partículas é menor.

No teste de escalabilidade fraca, o número de partículas por núcleo de processamento é prescrito em torno de 80.000. Modelos computacionais distintos incluindo até 3.200.000 partículas foram simulados neste teste. O tempo de processamento por passo de tempo foi adotado visando avaliar a performance paralela para estes cenários. A Figura 119 (b) mostra que a eficiência paralela para todos os métodos híbridos testados degrada com o aumento do tamanho do modelo. No entanto, esta degradação é mais lenta para o método baseado em HSFC.

11 Conclusão

Neste trabalho foram discutidos alguns aspectos relacionados a técnicas de computação de alto desempenho aplicadas ao Método dos Elementos Discretos. Algumas estratégias e aplicações de sistemas paralelos disponíveis na literatura foram apresentadas no intuito de embasar futuros trabalhos relacionados ao tema. O software adotado nos desenvolvimento foi o DEMOOP, sistema que vem recebendo contínuo aprimoramento desde 2005 no Laboratório de Computação Científica e Visualização da Universidade Federal de Alagoas. Este código computacional compõe o sistema computacional PetroDEM, desenvolvido com o intuito de auxiliar o entendimento de problemas da engenharia envolvendo meios descontínuos e com diversas aplicações de interesse da indústria de petróleo e gás.

Todos os desenvolvimentos realizados partem do estudo do *profile* do *software* DEMOOP, realizado com o objetivo de permitir otimizações em sua implementação computacional. Verificou-se que as tarefas relacionadas ao acesso à estrutura de dados de contato na implementação que utiliza a estrutura *hash table* impõem restrições ao processamento de modelos de larga escala. Uma implementação computacional de um estrutura de dados alternativa foi desenvolvida e o mesmo estudo de *profile* do código foi realizado para a nova implementação. Observa-se que a implementação computacional que faz uso da estrutura de dados baseada em matriz esparsa é mais eficiente para o modelo numérico apresentado. Esta implementação também permite um melhor controle de acesso à memória, uma vez que todos os dados das partículas estão armazenados de forma estruturada em basicamente um único vetor. Isto permite que o *software* ordene estes elementos de uma forma que explore melhor a localidade dos dados e permite um melhor uso da hierarquia de memória *cache*. A nova estratégia de acesso à estrutura de dados de contato de partículas foi incorporada ao código-fonte do *software* DEMOOP.

Diversos estudos de performance foram realizados em seguida visando otimizar o processamento paralelo de modelos do DEM em larga escala. As metodologias desenvolvidas, em geral, estão associadas com as técnicas de particionamento utilizadas e os ambientes de *hardware* adotados como referência. Adota-se, inicialmente, o modelo de programação paralelo baseado em troca de mensagens. Este modelo permite que uma simulação do DEM possa ser realizada tanto em ambientes computacionais de memória distribuída, como também em ambientes de memória compartilhada. Isto permite que um modelo numérico de larga escala possa ser simulado utilizando recursos computacionais de *clusters* de computadores ou mesmo em processadores que dispõem de mais de um núcleo de processamento. Este investimento de tempo de programação adicional justifica-se em virtude dos ganhos obtidos em tempos de computação, uma vez que as simulações dos modelos numéricos podem ser realizadas de forma mais rápida. Além disso, a implementação

permite que os modelos cresçam em escala, ocupando a memória de recursos computacionais distribuídos, muito além do que um único computador poderia armazenar.

O modelo inicial de particionamento de domínio é o de mais fácil implementação. Consiste em dividir o domínio em faixas horizontais ou verticais. Os subdomínios gerados possuem, portanto, vizinhanças conhecidas e a topologia de comunicação entre processos não varia ao longo da simulação. O desempenho de execução paralelo para esta técnica foi avaliado através de dois cenários que privilegiam ou não a carga de trabalho por núcleo de processamento. Os testes foram realizados usando, tanto modelos de distribuição de tarefas baseados em memória distribuída, como em memória compartilhada. Verifica-se que o cenário de geometria com uma dimensão dominante conduz a melhores performances de desempenho paralelo para a estratégia de decomposição geométrica por faixas. Foram obtidos valores de *speedup* da ordem de dezenas o que pode tornar viável o estudo de modelos de interesse cada vez mais complexos envolvendo elementos discretos. Também foi possível simular um modelo de larga escala (123 milhões de partículas) e verificar que o tempo de comunicação entre processos é o principal limitante para o estudo de modelos de grande porte utilizando esta metodologia. Embora o estudo tenha relatado uma boa aplicabilidade do cenário de simulação ao modelo de decomposição e solução do domínio, nem sempre a estratégia de divisão por faixas é apropriada.

Na sequência, foram desenvolvidas metodologias de particionamento mais gerais, onde a topologia de comunicação pode inclusive variar ao longo do tempo de simulação. Para tanto, é necessário o desenvolvimento de mecanismos que permitam a definição de vizinhança de subdomínios, ou seja, a forma como determinar quais partículas devem ser compartilhadas entre os processos para que o cálculo de forças interativas possam ocorrer de forma adequada. Uma vez que a malha que define a conectividade entre elementos não é montada explicitamente, já que o DEM é um método sem malha, a determinação da vizinhança não é um procedimento direto. Neste trabalho foram apresentadas três técnicas distintas que permitem a definição de vizinhança entre subdomínios: a técnica baseada em compartilhamento de matrizes de presença e duas técnicas combinadas (geometria + topologia) que utilizam malhas regulares auxiliares.

Apresentaram-se resultados referentes a distintas metodologias de particionamento, dentre elas o particionamento geométrico baseado em curvas de preenchimento de espaço e particionamentos multi-nível baseados em topologia e em um método que combina geometria e topologia. As decomposições obtidas para os particionamentos baseados em topologia apresentaram melhores resultados em termos de qualidade de particionamento. Verifica-se também a necessidade do emprego de técnicas de balanceamento dinâmico de carga quando há uma variação do domínio decorrente do movimento de partículas. A técnica de balanceamento de carga empregada, baseada no *software* Zoltan mostrou-se eficiente para a decomposição do domínio, inclusive nos casos onde há grandes movimentações

de partículas. Entretanto, deve-se atentar, quando em seu uso, para o parâmetro que pondera a função objetivo minimizada. Observa-se que quanto maior a importância que se dá à redução do número de cortes de arestas do grafo maior é o volume de comunicação necessário para a migração de partículas entre subdomínios.

Os estudos de performance realizados para as estratégias de paralelização baseados em topologia indicam que os mecanismos de comunicação utilizados na definição de vizinhança degradam consideravelmente o desempenho computacional. Isto ocorre de forma mais clara para a metodologia que utiliza o compartilhamento de matrizes de presença. Embora os métodos de particionamento topológicos criem boas decomposições de domínio com reduzidas interfaces entre subdomínios, a utilização destas decomposições requer procedimentos adicionais de determinação de vizinhos que são custosos e não escaláveis. Problema semelhante ocorre para as decomposições de domínio baseadas em curvas de preenchimento de espaço. Apesar de esta técnica ser baseada em mapeamentos puramente geométricos, não existe uma definição simples para os contornos geométricos dos subdomínios.

As técnicas geométricas de decomposição de domínio são, portanto, mais aplicáveis ao DEM em virtude da natureza descontínua do método e da alta mobilidade dos elementos, em especial nas simulações de fluxo de partículas. O método de decomposição baseado em RCB apresenta portanto uma melhor aplicabilidade ao DEM, o que pôde ser observado nos estudos de desempenho conduzidos nesta tese e na literatura até então consultada. Assim como outros métodos geométricos de decomposição, esta técnica cria células regulares que auxiliam o processo de determinação de vizinhança de processos. No caso específico, a quantidade de células geradas é igual ao número de subdomínios criados. As partições geradas tendem a ser incrementais, indicando que reparticionamentos sucessivos não alteram significativamente as partículas presentes nos subdomínios. Além disso os volumes de comunicação decorrentes dos procedimentos de particionamento também são reduzidos.

Em decorrência das experiências observadas para as estratégias de paralelização analisadas, uma metodologia geral de paralelização foi apresentada para as soluções de modelos de elementos discretos. A metodologia, descrita na forma de um fluxo de trabalho (*workflow*), envolve as tarefas de particionamento de domínio, balanço de carga, mecanismos de controle de memória, além do procedimento padrão de cálculo do DEM. Adicionalmente, uma estratégia de paralelização voltada para arquiteturas computacionais de memória compartilhada é introduzido. Esta estratégia permite que a solução dos subdomínios gerados pelo processo de decomposição possa ser realizada com o auxílio de *threads*. Desta forma, a metodologia desenvolvida segue uma estratégia de paralelização híbrido que considera o uso, tanto de recursos de paralelização por troca de mensagens, como de *threads*.

Da metodologia de solução apresentada é possível derivar diferentes estratégias de

solução paralela híbrida. Estas estratégias podem considerar um conjunto de algoritmos e técnicas com finalidades distintas. Neste trabalho foram apresentados três destas estratégias de solução. As estratégias diferem entre si basicamente na forma de particionamento local. A primeira técnica considera o particionamento local através do método RCB, assim como proposto no trabalho de Berger et al. (2015). A segunda técnica utiliza um método topológico de particionamento e, finalmente, a técnica proposta neste trabalho é concebida. Esta técnica baseia-se em procedimentos de mapeamento por curvas de Hilbert, combinando atividades de otimização de uso de memória, particionamento local e balanceamento de carga. A ideia principal do método é explorar as propriedades de localidade das curvas de Hilbert visando melhores particionamentos locais e maiores eficiências paralelas.

As implementações computacionais desenvolvidas são também incorporadas ao *software* DEMOOP e testes de eficiência são conduzidos com o intuito de avaliar as estratégias de solução. Os casos de estudo avaliados permitem analisar o *profile* do código computacional na sua nova configuração, além de métricas de eficiência, balanço de carga e escalabilidade.

Os experimentos numéricos mostram que a técnica híbrida baseada em particionamento de grafos reduz significativamente o número de testes geométricos nas interfaces entre *threads*, mas não apresenta um bom balanço de carga. Em contraste, os métodos de particionamento local baseados em RCB e HSFC apresentam maiores interfaces entre *threads* mas um melhor controle de balanço de carga. Uma vez que o método baseado em HSFC acarreta menores custos computacionais na definição das partições locais, o mesmo acaba obtendo um desempenho superior aos apresentados pelos outros métodos considerados. Os testes de escalabilidade forte e fraco também relatam boas performances paralelas, em especial para as configurações de execução híbridas. Isto ocorre uma vez que estas formas de execução permitem a utilização de um maior número de subconjuntos de partículas, menores números de subdomínios e consequentemente menos comunicações entre processos.

11.1 Sugestões para trabalhos futuros

Embora diversos aspectos tenham sido tratados nesta tese quanto à paralelização do DEM em larga escala, várias metodologias ainda podem tratadas com o intuito de avaliar e melhorar os resultados obtidos. Como proposta inicial sugere-se o estudo de metodologias que melhor definam a vizinhança de partículas em modelos distribuídos entre processos. Conforme demonstrado, a técnica baseada em comunicação de matrizes de presença não é eficiente quando o número de processos cresce.

Outra questão que se pode levantar é quanto ao uso de mecanismos de ordenação de partículas em memória para as técnicas de particionamento híbrido. Sugere-se investigar

a utilização de estruturas de dados que armazenem o conjunto de partículas em subconjuntos. Isto pode ser feito através da separação do vetor de partículas em vetores menores atribuídos às *threads*. Desta forma, uma partícula pode migrar de um vetor para o outro durante a simulação sem que haja a necessidade de ordenação. Cabe a ressalva que este procedimento pode afetar todo o funcionamento do código computacional já que a estrutura de dados muda significativamente.

Por fim, sugere-se o estudo de mecanismos híbridos de paralelização que considerem ambientes computacionais heterogêneos. Isso pode ser realizados através da combinação de processamento em CPUs, GPUs e aceleradores. Neste modelo de desenvolvimento, por exemplo, processos MPI dividem a carga computacional para nós de processamento e, dentro de cada nó, a solução do DEM pode ser executada em sistemas complexos contendo diversas arquiteturas de *hardware*. A computação em *hardware* heterogêneo pode ser realizada através da API de programação OpenCL em dispositivos compatíveis. Dentre as vantagens deste modelo de programação está a possibilidade de uso de processamento acelerado em placas gráficas.

Referências

ABE, S.; GENT, H. van; URAI, J. L. Dem simulation of normal faults in cohesive materials. *Tectonophysics*, v. 512, n. 1-4, p. 12–21, nov. 2011. Citado na página 19.

ABE, S.; PLACE, D.; MORA, P. A parallel implementation of the lattice solid model for the simulation of rock mechanics and earthquake dynamics. *Pure and Applied Geophysics*, 161, n. 11-12, p. 2265–2277, DEC 2004. ISSN 0033-4553. Citado na página 20.

AHRENS, J.; GEVECI, B.; LAW, C. Paraview: An end-user tool for large-data visualization. *The Visualization Handbook*, Citeseer, p. 717, 2005. Citado 2 vezes nas páginas 29 e 85.

AHRENS, J.; GEVECI, B.; LAW, C. Visualization handbook. In: _____. [S.l.]: Academic Press, 2005. cap. ParaView: An End-User Tool for Large Data Visualization., p. 717–732. Citado 2 vezes nas páginas 73 e 93.

ALEXIADES, V.; AMIEZ, G.; GREMAUD, P. A. Super-time-stepping acceleration of explicit schemes for parabolic problems. *Communications In Numerical Methods In Engineering*, v. 12, n. 1, p. 31–42, 1996. Citado na página 20.

AMORIM, J. A.; CINTRA, D. T.; LIRA, W. W. M. Uma interface gráfica-interativa para visualização dinâmica do método dos elementos discretos. In: *XXVII Iberian Latin American Congress On Computational Methods In Engineering*. Belém/PA-Brasil: [s.n.], 2006. p. 360–360. Citado 3 vezes nas páginas 29, 73 e 93.

BADER, M. *Space-filling curves: an introduction with applications in scientific computing*. [S.l.]: Springer Science & Business Media, 2012. Citado na página 21.

BARNARD, S. T.; SIMON, H. *A Parallel Implementation of Multilevel Recursive Spectral Bisection For Application To Adaptive Unstructured Meshes*. [S.l.]: Siam, 1995. 627–632 p. Citado na página 64.

BELYTSCHKO, T.; HUGHES, T. J. R. *Computational Methods For Transient Analysis*. Amsterdam: North-Holland, 1983. Citado 2 vezes nas páginas 51 e 53.

BERGER, M. J.; BOKHARI, S. H. A partitioning strategy for nonuniform problems on multiprocessors. *Ieee Transactions On Computers*, v. 36, n. 5, p. 570–580, maio 1987. Citado na página 56.

BERGER, R. et al. Hybrid parallelization of the liggghts open-source dem code. *Powder Technology*, Elsevier, v. 278, p. 234–247, 2015. Citado 7 vezes nas páginas 78, 102, 109, 110, 156, 160 e 170.

BOMAN, E. et al. *Zoltan 3.0: Parallel Partitioning, Load-balancing, and Data Management Services; User's Guide*. Albuquerque, NM, 2007. Tech. Report SAND2007-4748W. Citado na página 129.

BOMAN, E. G. et al. The Zoltan and Isorropia parallel toolkits for combinatorial scientific computing: Partitioning, ordering, and coloring. *Scientific Programming*, v. 20, n. 2, 2012. Citado 5 vezes nas páginas 29, 79, 88, 99 e 122.

BROWN, K. et al. Parallel strategies for crash and impact simulations. *Computer Methods in Applied Mechanics and Engineering*, v. 184, n. 2-4, p. 375–390, 2000. ISSN 0045-7825. Citado na página 19.

BUI, T. N.; MOON, B. R. Genetic algorithm and graph partitioning. *IEEE Trans. Comput.*, IEEE Computer Society, Washington, DC, USA, v. 45, n. 7, p. 841–855, jul. 1996. ISSN 0018-9340. Disponível em: <http://dx.doi.org/10.1109/12.508322>. Citado na página 62.

CAI, L.; YANG, X.; WANG, Y. All fourteen bravais lattices can be formed by interference of four noncoplanar beams. *Optics letters*, Optical Society of America, v. 27, n. 11, p. 900–902, 2002. Citado na página 36.

CALLISTER, W. D.; RETHWISCH, D. G. *Fundamentals of materials science and engineering*. [S.l.]: Wiley, 2013. Citado na página 37.

CAMPBELL, C. S.; CLEARY, P. W.; HOPKINS, M. Large-scale landslide simulations: Global deformation, velocities and basal friction. *Journal of Geophysical Research: Solid Earth (1978–2012)*, Wiley Online Library, v. 100, n. B5, p. 8267–8283, 1995. Citado 2 vezes nas páginas 19 e 133.

CARVALHO, M. T. M.; MARTHA, L. F.; CELES, W. Pos3d: um pósprocessador genérico para modelos 3d de elementos finitos. In: *Anais do X Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens*. São Paulo - Brasil: [s.n.], 1997. p. 10–14. Citado na página 72.

CHANG, K.-J.; TABOADA, A. Discrete element simulation of the Jiufengershan rock-and-soil avalanche triggered by the 1999 Chi-Chi earthquake, Taiwan. *Journal of Geophysical Research-Earth Surface*, 114, JUL 22 2009. ISSN 0148-0227. Citado 2 vezes nas páginas 19 e 20.

CHAPMAN, B.; JOST, G.; PAS, R. van der. *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*. [S.l.]: The MIT Press, 2007. ISBN 0262533022, 9780262533027. Citado na página 20.

CHEN, F. et al. Multi-scale hpc system for multi-scale discrete simulation-development and application of a supercomputer with 1 petaflops peak performance in single precision. *Particuology*, v. 7, n. 4, p. Sino Greman Ctr Res Promot; Natl Nat Sci Fdn China; German Res Fdn, ago. 2009. Citado 2 vezes nas páginas 20 e 102.

CINTRA, D. T. et al. Aplicação de um framework de integração temporal adaptativa a problemas de fratura coesiva. In: *XXIX Iberian Latin American Congress on Computational Methods in Engineering*. Maceió-AL: [s.n.], 2008. Citado na página 20.

CUI, L.; O'SULLIVAN, C. Analysis of a triangulation based approach for specimen generation for discrete element simulations. *Granular Matter*, v. 5, n. 3, p. 135–145, dez. 2003. Citado na página 32.

CUNDALL, P. A.; HART, R. D. Numerical modelling of discontinua. *Engineering computations*, MCB UP Ltd, v. 9, n. 2, p. 101–113, 1992. Citado na página 32.

CUNDALL, P. A.; STRACK, O. D. L. Discrete numerical-model for granular assemblies. *Geotechnique*, v. 29, n. 1, p. 47–65, 1979. Citado 3 vezes nas páginas 19, 32 e 38.

DAGUM, L.; MENON, R. Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, IEEE, v. 5, n. 1, p. 46–55, 1998. Citado na página 104.

DRIESSCHE, R. V.; ROOSE, D. Dynamic load balancing with a spectral bisection algorithm for the constrained graph partitioning problem. In: SPRINGER. *High-Performance Computing and Networking*. [S.l.], 1995. p. 392–397. Citado na página 68.

FAHMY, M. W.; NAMINI, A. H. A survey of parallel nonlinear dynamic analysis methodologies. *Computers & Structures*, v. 53, n. 4, p. 1033–1043, nov. 1994. Citado na página 51.

FATTEBERT, J. . L.; RICHARDS, D. F.; GLOSLI, J. N. Dynamic load balancing algorithm for molecular dynamics based on voronoi cells domain decompositions. *Computer Physics Communications*, v. 183, n. 12, p. 2608–2615, dez. 2012. Citado na página 26.

FENG, Y.; OWEN, D. An augmented spatial digital tree algorithm for contact detection in computational mechanics. *International Journal For Numerical Methods In Engineering*, 55, n. 2, p. 159–176, SEP 20 2002. ISSN 0029-5981. Citado na página 40.

FENG, Y. T.; HAN, K.; OWEN, D. R. J. Filling domains with disks: an advancing front approach. *International Journal For Numerical Methods In Engineering*, v. 56, n. 5, p. 699–713, fev. 2003. Citado na página 33.

FERRAZ, D. H. S. et al. Um sistema gráfico-interativo para geração de modelos particulados. In: *XXIX CILAMCE - Congresso Ibero Latino Americano de Métodos Computacionais em Engenharia*. Maceió/AL-Brasil: [s.n.], 2008. Citado na página 72.

FLEISSNER, F.; EBERHARD, P. Dynamical particle simulation with parallel cache-aware domain decomposition strategies. *PAMM*, Wiley Online Library, v. 5, n. 1, p. 657–658, 2005. Citado na página 78.

FLEISSNER, F.; EBERHARD, P. Parallel load-balanced simulation for short-range interaction particle methods with hierarchical particle grouping based on orthogonal recursive bisection. *International Journal for Numerical Methods in Engineering*, Wiley Online Library, v. 74, n. 4, p. 531–553, 2008. Citado 2 vezes nas páginas 25 e 26.

FORM, W.; ITO, N.; KOHRING, G. A. Vectorized and parallelized algorithms for multimillion particle md-simulation. *International Journal of Modern Physics C-physics and Computers*, v. 4, n. 6, p. 1085–1101, dez. 1993. Citado na página 21.

FRERY, A. C. et al. Stochastic particle packing with specified granulometry and porosity. *Granular Matter*, Springer, v. 14, n. 1, p. 27–36, 2012. Citado na página 77.

GRAHAM, S. L.; KESSLER, P. B.; MCKUSICK, M. K. Gprof: A call graph execution profiler. *SIGPLAN Not.*, ACM, New York, NY, USA, v. 39, n. 4, p. 49–57, abr. 2004. ISSN 0362-1340. Disponível em: <http://doi.acm.org/10.1145/989393.989401>. Citado na página 73.

GREST, G. S.; DUNWEG, B.; KREMER, K. Vectorized link cell fortran code for molecular-dynamics simulations for a large number of particles. *Computer Physics Communications*, v. 55, n. 3, p. 269–285, out. 1989. Citado 2 vezes nas páginas 21 e 23.

GROUP, I. I. C. *PFC – Particle Flow Code, Ver. 5.0*. Minneapolis: [s.n.], 2014. User's guide. Citado na página 134.

HAN, H.; TSENG, C.-W. Exploiting locality for irregular scientific codes. *Parallel and Distributed Systems, IEEE Transactions on*, IEEE, v. 17, n. 7, p. 606–618, 2006. Citado na página 77.

HAN, K.; FENG, Y. T.; OWEN, D. R. J. Sphere packing with a geometric based compression algorithm. *Powder Technology*, v. 155, n. 1, p. 33–41, jul. 2005. Citado na página 33.

HAN, K.; FENG, Y. T.; OWEN, D. R. J. Performance comparisons of tree-based and cell-based contact detection algorithms. *Engineering Computations*, EMERALD GROUP PUBLISHING LIMITED, 60/62 TOLLER LANE, BRADFORD BD8 9BY, W YORKSHIRE, ENGLAND, 24, n. 1-2, p. 165–181, 2007. ISSN 0264-4401. Citado na página 40.

HENDRICKSON, B.; DEVINE, K. Dynamic load balancing in computational mechanics. *Computer Methods In Applied Mechanics and Engineering*, v. 184, n. 2-4, p. 485–500, 2000. Citado 2 vezes nas páginas 25 e 57.

HENDRICKSON, B.; LELAND, R. A multilevel algorithm for partitioning graphs. *Supercomputing '95, Proceedings, Vols 1 and 2*, p. Assoc Comp Machinery Inc; IEEE Comp Soc, 1995. Citado na página 63.

HENTY, D. S. Performance of hybrid message-passing and shared-memory parallelism for discrete element modeling. In: IEEE COMPUTER SOCIETY. *Proceedings of the 2000 ACM/IEEE conference on Supercomputing*. [S.l.], 2000. p. 10. Citado na página 102.

HILBERT, D. Ueber die stetige Abbildung einer Line auf ein Flächenstück. *Mathematische Annalen*, v. 38, n. 3, p. 459–460, 1891. Disponível em: <http://dx.doi.org/10.1007/bf01199431>. Citado 2 vezes nas páginas 28 e 79.

HU, Y. F.; BLAKE, R. J. Progress in computer research. In: COLUMBUS, F. (Ed.). Commack, NY, USA: Nova Science Publishers, Inc., 2001. cap. Load Balancing for Unstructured Mesh Applications, p. 117–148. ISBN 1-59033-011-0. Citado 5 vezes nas páginas 57, 60, 61, 63 e 64.

HUGHES, C.; HUGHES, T. *Parallel and Distributed Programming Using C++*. [S.l.]: Prentice Hall Professional Technical Reference, 2003. ISBN 0131013769. Citado na página 20.

HUGHES, T. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Englewood, New Jersey: Prentice-Hall, 1987. Citado na página 51.

JONES, M. T.; PLASSMANN, P. E. Computational results for parallel unstructured mesh computations. *Computing Systems In Engineering*, v. 5, n. 4-6, p. 297–309, ago. 1994. Citado na página 56.

JÚNIOR, H. C.; CINTRA, D. T.; SILVEIRA, E. S. S. Desenvolvimento de um sistema orientado a objetos para análise através do método dos elementos discretos. In: *XXVII Latin American Congress on Computational Methods in Engineering, 2006*. Belém/PA-Brasil: [s.n.], 2006. p. 488–488. Citado 2 vezes nas páginas 28 e 72.

KACIANAUSKAS, R. et al. Parallel discrete element simulation of poly-dispersed granular material. *Advances in Engineering Software*, ELSEVIER SCI LTD, THE BOULEVARD, LANGFORD LANE, KIDLINGTON, OXFORD OX5 1GB, OXON, ENGLAND, 41, n. 1, p. 52–63, JAN 2010. ISSN 0965-9978. Citado na página 23.

KANJI, M. A.; CRUZ, P. T.; MASSAD, F. Debris flow affecting the cubatão oil refinery, brazil. *Landslides*, Springer, v. 5, n. 1, p. 71–82, 2008. Citado na página 133.

KARYPIS, G.; KUMAR, V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, SIAM, v. 20, n. 1, p. 359–392, 1998. Citado 2 vezes nas páginas 29 e 111.

KARYPIS, G.; KUMAR, V. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, ACADEMIC PRESS INC, 525 B ST, STE 1900, SAN DIEGO, CA 92101-4495 USA, v. 48, n. 1, p. 96–129, JAN 1998. ISSN 0743-7315. Citado 6 vezes nas páginas 58, 59, 62, 63, 64 e 122.

KARYPIS, G.; KUMAR, V. Parallel multilevel k-way partitioning scheme for irregular graphs. *Siam Review*, Siam Publications, v. 41, n. 2, p. 278–300, jun. 1999. Citado 2 vezes nas páginas 28 e 64.

KARYPIS, G.; SCHLOEGEL, K.; KUMAR, V. *ParMeTiS: Parallel graph partitioning and sparse matrix ordering library–version 3.1. University of Minnesota*. [S.l.], 2003. Citado 3 vezes nas páginas 29, 58 e 62.

KERNIGHAN, B. W.; LIN, S. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell system technical journal*, v. 49, n. 1, p. 291–307, 1970. Citado 2 vezes nas páginas 61 e 62.

KNILL, J. Report on the shum wan road landslide of 13 august 1995. *Independent Review of the Investigation by the Geotechnical Engineering Office. Geotechnical Engineering Office, Hong Kong*, 1996. Citado 3 vezes nas páginas 143, 144 e 152.

KOHRING, G. A. Dynamic load balancing for parallelized particle simulations on mimd computers. *Parallel Computing*, v. 21, n. 4, p. 683–693, abr. 1995. Citado 4 vezes nas páginas 22, 23, 26 e 56.

KORADI, R.; BILLETER, M.; GÜNTERT, P. Point-centered domain decomposition for parallel molecular dynamics simulation. *Computer Physics Communications*, Elsevier, v. 124, n. 2, p. 139–147, 2000. Citado na página 26.

KRYSL, P.; BELYTSCHKO, T. Object-oriented parallelization of explicit structural dynamics with PVM. *Computers & Structures*, v. 66, n. 2-3, p. 259–273, jan. 1998. Citado na página 52.

KUNASETH, M. et al. Analysis of scalable data-privatization threading algorithms for hybrid mpi/openmp parallelization of molecular dynamics. *Journal of Supercomputing*, v. 66, n. 1, p. 406–430, out. 2013. Citado na página 102.

LABRA, C.; ONATE, E. High-density sphere packing for discrete element method simulations. *Communications In Numerical Methods In Engineering*, v. 25, n. 7, p. 837–849, jul. 2009. Citado na página 33.

LAGES, E. N. et al. An adaptive time integration strategy based on displacement history curvature. *International Journal For Numerical Methods In Engineering*, WILEY-BLACKWELL, 111 RIVER ST, HOBOKEN 07030-5774, NJ USA, v. 93, n. 12, p. 1235–1254, MAR 23 2013. ISSN 0029-5981. Citado 2 vezes nas páginas 20 e 50.

LAWDER, J. K.; KING, P. J. Using state diagrams for Hilbert curve mappings. *International Journal of Computer Mathematics*, Taylor & Francis, v. 78, n. 3, p. 327–342, 2001. Citado na página 79.

LEVENBERG, K. A method for the solution of certain problems in least squares. *Quart. Applied Math.*, v. 2, p. 164–168, 1944. Citado 2 vezes nas páginas 31 e 35.

LI, W. C. et al. Discrete element modeling of a rainfall-induced flowslide. *Engineering Geology*, Elsevier, v. 149, p. 22–34, 2012. Citado 4 vezes nas páginas 11, 133, 134 e 137.

LIU, L.; ZHANG, Z.; YU, A. Dynamic simulation of the centripetal packing of mono-sized spheres. *Physica A*, v. 268, n. 3-4, p. 433–453, JUN 15 1999. ISSN 0378-4371. Citado na página 37.

LOPES, E. S.; RIEDEL, P. S. *Simulação de corrida de detritos na bacia do Rio das Pedras que afetou a Refinaria Presidente Bernardes em Cubatão -SP*. 2007. Disponível em: <http://urlib.net/rep/sid.inpe.br/mtc-m17@80/2007/06.28.12.48>. Citado na página 132.

LOWRIE, R. B. A comparison of implicit time integration methods for nonlinear relaxation and diffusion. *Journal of Computational Physics*, v. 196, n. 2, p. 566–590, maio 2004. Citado na página 51.

LUKAS, T.; D'ALBANO, G. S.; MUNJIZA, A. Space decomposition based parallelization solutions for the combined finite–discrete element method in 2d. *Journal of Rock Mechanics and Geotechnical Engineering*, Elsevier, v. 6, n. 6, p. 607–615, 2014. Citado na página 25.

MAKNICKAS, A. et al. Parallel DEM software for simulation of granular media. *Informatica*, IOS Press, v. 17, n. 2, p. 207–224, 2006. Citado 3 vezes nas páginas 20, 23 e 24.

MARKAUSKAS, D.; KAČENIAUSKAS, A. The comparison of two domain repartitioning methods used for parallel discrete element computations of the hopper discharge. *Advances in Engineering Software*, v. 84, n. 0, p. 68–76, 2015. ISSN 0965-9978. Citado 3 vezes nas páginas 28, 156 e 164.

MARKAUSKAS, D.; KACENIAUSKAS, A.; MAKNICKAS, A. Dynamic domain decomposition applied to hopper discharge simulations by discrete element method. *Information Technology and Control*, KAUNAS UNIV TECHNOLOGY, KAUNAS UNIV TECHNOL, DEPT ELECTRONICS ENGINEERING, STUDENTU STR 50, KAUNAS, LT-51368, LITHUANIA, 40, n. 4, p. 286–292, 2011. ISSN 1392-124X. Citado 2 vezes nas páginas 26 e 27.

MARQUARDT, D. W. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, SIAM, v. 11, n. 2, p. 431–441, 1963. Citado 2 vezes nas páginas 31 e 35.

MASSAD, F. et al. Characteristics and volume of sediments transport in debris flows in serra do mar, cubatão, brazil, int. In: *Workshop on the Debris Flow Disaster of Dec.* [S.l.: s.n.], 1999. Citado na página 133.

MCCOOL, M. D.; ROBISON, A. D.; REINDERS, J. Structured parallel programming: patterns for efficient computation. Elsevier, 2012. Citado na página 106.

MENEZES, I. F.; CARVALHO, M. T. M.; MIRANDA, A. C. de O. *Neutral File*. 2002. Http://www.tecgraf.puc-rio.br/neutralfile/. Acessado em: 21/11/2013. Citado na página 72.

MOON, B. et al. Analysis of the clustering properties of the hilbert space-filling curve. *Knowledge and Data Engineering, IEEE Transactions on*, IEEE, v. 13, n. 1, p. 124–141, 2001. Citado na página 79.

MUNJIZA, A. *The Combined Finite-Discrete Element Method*. [S.l.]: Wiley, 2004. Hardcover. ISBN 0470841990. Citado na página 31.

MUNJIZA, A.; ANDREWS, K. R. F. Nbs contact detection algorithm for bodies of similar size. *International Journal for Numerical Methods in Engineering*, Department of Engineering, QMW, University of London, London, U.K., v. 43, n. 1, p. 131–149, 1998. ISSN 1097-0207. Citado 3 vezes nas páginas 40, 41 e 43.

MUNJIZA, A.; CLEARY, P. W. Industrial particle flow modelling using discrete element method. *Engineering Computations*, Emerald Group Publishing Limited, v. 26, n. 6, p. 698–743, 2009. Citado na página 133.

NEWMARK, N. M. A method of computation for structural dynamics. *ASCE Journal of Engineering Mechanics Division*, v. 85, n. 3, p. 67–94, 1959. Citado 2 vezes nas páginas 9 e 53.

NGUYEN, H. *GPU Gems 3*. [S.l.]: Addison-Wesley Professional, 2007. ISBN 9780321515261. Citado na página 20.

NISHIURA, D.; SAKAGUCHI, H. Parallel-vector algorithms for particle simulations on shared-memory multiprocessors. *Journal of Computational Physics*, v. 230, n. 5, p. 1923–1938, mar. 2011. Citado 2 vezes nas páginas 20 e 102.

ONATE, E.; ROJEK, J. Combination of discrete element and finite element methods for dynamic analysis of geomechanics problems. *Computer methods in applied mechanics and engineering*, Elsevier, v. 193, n. 27, p. 3087–3128, 2004. Citado 2 vezes nas páginas 19 e 47.

O'SULLIVAN, C.; BRAY, J. D. Selecting a suitable time step for discrete element simulations that use the central difference time integration scheme. *Engineering Computations*, v. 21, n. 2-4, p. 278–303, 2004. Citado 3 vezes nas páginas 53, 54 e 85.

OU, C.-W.; RANKA, S. Parallel incremental graph partitioning. *Parallel and Distributed Systems, IEEE Transactions on*, IEEE, v. 8, n. 8, p. 884–896, 1997. Citado na página 70.

OWEN, D. et al. The modelling of multi-fracturing solids and particulate media. *International Journal for Numerical Methods in Engineering*, Wiley Online Library, v. 60, n. 1, p. 317–339, 2004. Citado 3 vezes nas páginas 19, 26 e 27.

OWEN, D. R. J.; FENG, Y. T. Parallelised finite/discrete element simulation of multi-fracturing solids and discrete systems. *Engineering Computations*, v. 18, n. 3-4, p. 557–576, 2001. Citado 4 vezes nas páginas 26, 27, 55 e 65.

OWENS, J. D. et al. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26, n. 1, p. 80–113, 2007. ISSN 0167-7055. Citado na página 20.

PARK, K. C.; UNDERWOOD, P. G. A variable-step central difference method for structural dynamics analysis -part 1: Theoretical aspects. *Computer Methods in Applied Mechanics and Engineering*, v. 22, n. 2, p. 241–258, 1980. Citado na página 52.

PERKINS, E.; WILLIAMS, J. R. A fast contact detection algorithm insensitive to object sizes. *Engineering Computations*, MCB UP Ltd, v. 18, n. 1/2, p. 48–62, 2001. Citado 3 vezes nas páginas 40, 41 e 45.

PETRODEM. 2013. Http://petrodem.lccv.ufal.br/. Acessado em: 25/06/2013. Citado 2 vezes nas páginas 72 e 73.

POTHEN, A.; SIMON, H. D.; LIOU, K. P. Partitioning sparse matrices with eigenvectors of graphs. *Siam Journal On Matrix Analysis and Applications*, Siam Publications, v. 11, n. 3, p. 430–452, jul. 1990. Citado na página 60.

POTYONDY, D.; CUNDALL, P. A bonded-particle model for rock. *International Journal of Rock Mechanics and Mining Sciences*, 41, n. 8, p. 1329–1364, DEC 2004. ISSN 1365-1609. Citado na página 31.

RADEKE, C. A.; GLASSER, B. J.; KHINAST, J. G. Large-scale powder mixer simulations using massively parallel gpu architectures. *Chemical Engineering Science*, v. 65, n. 24, p. 6435–6442, dez. 2010. Citado na página 20.

RAPAPORT, D. C. Large-scale molecular dynamics simulation using vector and parallel computers. *Computer Physics Reports*, ELSEVIER SCIENCE BV, PO BOX 211, 1000 AE AMSTERDAM, NETHERLANDS, 9, n. 1, p. 1–53, DEC 1988. ISSN 0167-7977. Citado na página 21.

RYCROFT, C. H. et al. Dynamics of random packings in granular flow. *Physical Review E*, v. 73, n. 5, p. 051306, maio 2006. Citado na página 19.

SAKAI, M. et al. Large-scale discrete element modeling in a fluidized bed. *International Journal For Numerical Methods In Fluids*, v. 64, n. 10-12, p. 1319–1335, dez. 2010. Citado na página 19.

SBALZARINI, I. F. et al. Ppm - a highly efficient parallel particle-mesh library for the simulation of continuum systems. *Journal of Computational Physics*, v. 215, n. 2, p. 566–588, jul. 2006. Citado 2 vezes nas páginas 20 e 25.

SBALZARINI, I. F. et al. A software framework for the portable parallelization of particle-mesh simulations. *Euro-par 2006 Parallel Processing*, v. 4128, p. 730–739, 2006. Citado na página 25.

SCHLOEGEL, K.; KARYPIS, G.; KUMAR, V. Multilevel diffusion schemes for repartitioning of adaptive meshes. *Journal of Parallel and Distributed Computing*, Elsevier, v. 47, n. 2, p. 109–124, 1997. Citado na página 70.

SCHLOEGEL, K.; KARYPIS, G.; KUMAR, V. Wavefront diffusion and lmsr: Algorithms for dynamic repartitioning of adaptive meshes. *IEEE Transactions on Parallel and Distributed Systems*, IEEE, v. 12, n. 5, p. 451–466, 2001. Citado na página 68.

SENA, M. C. R.; SILVEIRA, E. S. S. Uma implementação do método dos elementos discretos em um ambiente de computação paralela. In: *XXVIII Latin American Congress on Computational Methods in Engineering*. Porto-Portugal: [s.n.], 2007. Citado 2 vezes nas páginas 29 e 56.

SHIGETO, Y.; SAKAI, M. Parallel computing of discrete element method on multi-core processors. *Particuology*, v. 9, n. 4, p. 398–405, ago. 2011. Citado 2 vezes nas páginas 20 e 102.

SHOJAAEE, Z. et al. An adaptive hierarchical domain decomposition method for parallel contact dynamics simulations of granular materials. *Journal of Computational Physics*, v. 231, n. 2, p. 612 – 628, 2012. ISSN 0021-9991. Citado na página 25.

SIIRIA, S.; YLIRUUSI, J. Particle packing simulations based on newtonian mechanics. *Powder Technology*, v. 174, n. 3, p. 82–92, MAY 25 2007. ISSN 0032-5910. Citado na página 37.

SIMON, H. D. Partitioning of Unstructured Problems for Parallel Processing. *Computing Systems in Engineering*, v. 2, p. 135–148, 1991. Disponível em: <http://citeseer.ist.psu.edu/simon94partitioning.html>. Citado na página 57.

SNIR, M. et al. *MPI-The Complete Reference, Volume 1: The MPI Core*. 2nd. (revised). ed. Cambridge, MA, USA: MIT Press, 1998. ISBN 0262692155. Citado 2 vezes nas páginas 101 e 104.

STIJNMAN, M. A.; BISSELING, R. H.; BARKEMA, G. T. Partitioning 3d space for parallel many-particle simulations. *Computer Physics Communications*, v. 149, n. 3, p. PII S0010–4655(02)00628–8, jan. 2003. Citado 3 vezes nas páginas 23, 24 e 56.

TANG, C.-L. et al. The tsaoling landslide triggered by the chi-chi earthquake, taiwan: insights from a discrete element simulation. *Engineering Geology*, Elsevier, v. 106, n. 1, p. 1–19, 2009. Citado na página 133.

THORNTON, C. Numerical simulations of deviatoric shear deformation of granular media. *Geotechnique*, v. 50, n. 1, p. 43–53, fev. 2000. Citado na página 54.

VENETILLO, J. S.; CELES, W. Gpu-based particle simulation with inter-collisions. *Vis. Comput.*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 23, n. 9, p. 851–860, 2007. ISSN 0178-2789. Citado na página 31.

WALSHAW, C.; CROSS, M.; EVERETT, M. Parallel dynamic graph partitioning for adaptive unstructured meshes. *Journal of Parallel and Distributed Computing*, Elsevier, v. 47, n. 2, p. 102–108, 1997. Citado na página 70.

WALTHER, J. H.; SBALZARINI, I. F. Large-scale parallel discrete element simulations of granular flow. *Engineering Computations*, v. 26, n. 6, p. 688–697, 2009. Citado 4 vezes nas páginas 19, 20, 25 e 117.

WANG, F.; FENG, Y.; OWEN, D. Interprocessor communication schemes in parallel finite-discrete element analysis on pc clusters. *Engineering Computations*, EMERALD GROUP PUBLISHING LIMITED, HOWARD HOUSE, WAGON LANE, BINGLEY BD16 1WA, W YORKSHIRE, ENGLAND, v. 20, n. 7-8, p. 1065–1084, 2003. ISSN 0264-4401. Citado na página 25.

WANG, F.; FENG, Y.; OWEN, D. Parallelisation for finite-discrete element analysis in a distributed-memory environment. *International Journal of Computational Engineering Science*, World Scientific, v. 5, n. 01, p. 1–23, 2004. Citado na página 28.

WANG, F. et al. Parallel analysis of combined finite/discrete element systems on PC cluster. *ACTA MECHANICA SINICA*, SPRINGER HEIDELBERG, TIERGARTENSTRASSE 17, D-69121 HEIDELBERG, GERMANY, 20, n. 5, p. 534–540, OCT 2004. ISSN 0567-7718. Citado 2 vezes nas páginas 25 e 26.

WANG, F.; SASSA, K. Landslide simulation by a geotechnical model combined with a model for apparent friction change. *Physics and Chemistry of the Earth, Parts A/B/C*, Elsevier, v. 35, n. 3, p. 149–161, 2010. Citado 2 vezes nas páginas 143 e 153.

WANG, Y. et al. Implementation of particle-scale rotation in the 3-d Lattice Solid Model. *Pure and Applied Geophysics*, 163, n. 9, p. 1769–1785, SEP 2006. ISSN 0033-4553. Citado na página 20.

WILLIAMS, J. R.; PERKINS, E.; COOK, B. A contact algorithm for partitioning arbitrary sized objects. *Engineering Computations: Int J for Computer-Aided Engineering*, Emerald Group Publishing Limited, p. 235–248, 2004. ISSN 0264-4401. Disponível em: <http://dx.doi.org/10.1108/02644400410519767>. Citado na página 41.

XDMF - eXtensible Data Model and Format. 2013. Http://www.xdmf.org/. Acessado em: 12/11/2013. Citado na página 73.

XU, J. et al. Quasi-real-time simulation of rotating drum using discrete element method with parallel GPU computing. *Particuology*, Elsevier, v. 9, n. 4, p. 446–450, 2011. Citado na página 20.

YAO, Z. et al. Improved neighbor list algorithm in molecular simulations using cell decomposition and data sorting method. *Computer physics communications*, Elsevier, v. 161, n. 1, p. 27–35, 2004. Citado na página 78.

ZHANG, D.; JIANG, C.; LI, S. A fast adaptive load balancing method for parallel particle-based simulations. *Simulation Modelling Practice and Theory*, v. 17, n. 6, p. 1032–1042, jul. 2009. Citado na página 25.