Pontifícia Universidade Católica
do Rio de Janeiro

André Luiz Ferreira Pinto

Algoritmo para geração de arranjos de partículas para utilização no método dos elementos discretos

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Engenharia Civil do Departamento de Engenharia Civil da PUC–Rio

Orientador: Prof. Deane de Mesquita Roehl

Rio de Janeiro
Agosto de 2009

Pontifícia Universidade Católica
do Rio de Janeiro

André Luiz Ferreira Pinto

Algoritmo para geração de arranjos de partículas para utilização no método dos elementos discretos

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Engenharia Civil do Departamento de Engenharia Civil do Centro Técnico Científico da PUC–Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Deane de Mesquita Roehl
Orientador
Departamento de Engenharia Civil — PUC–Rio

Prof. Elisa D. Sotelino
Virginia Polytechnic Institute and State University

Prof. Waldemar Celes Filho
Pontifícia Universidade Católica do Rio de Janeiro

Prof. Celso Romanel
Pontifícia Universidade Católica do Rio de Janeiro

Prof. José Eugenio Leal
Coordenador Setorial do Centro Técnico Científico — PUC–Rio

Rio de Janeiro, 14 de Agosto de 2009

André Luiz Ferreira Pinto

Graduado em Engenharia Civil pela Universidade do Estado do Rio de Janeiro em 2007. No mesmo ano iniciou o curso de Mestrado em Engenharia Civil na PUC–Rio, na área de concentração de Estruturas, atuando na linha de pesquisa do Método dos Elementos Discretos, sendo bolsista CNPq durante esse período.

Ficha Catalográfica

Pinto, André Luiz Ferreira

Algoritmo para geração de arranjos de partículas para utilização no método dos elementos discretos : / André Luiz Ferreira Pinto; orientador: Deane de Mesquita Roehl. — 2009.

77 f.: il.(color.) ; 30 cm

Dissertação (Mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2009.

Inclui bibliografia.

1. Engenharia Civil – Teses. 2. Método dos elementos discretos. 3. Otimização. 4. Algoritmo genético. 5. Material com gradação funcional. I. Roehl, Deane de Mesquita. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Civil. III. Título.

CDD: 624

Agradecimentos

Agradeço a todos que contribuíram direta ou indiretamente para a realização deste trabalho, amigos, colegas e professores, e em especial:

Aos meus pais, Luiz Alberto e Heloisa Helena, pelo apoio em todas as horas, pelo tempo que puderam: a educação esteve sempre em destaque. Por todo o carinho e incentivo, assim como o de meus irmãos — Carolina, Marcelo, Mariana e Bruno.

À professora Deane de Mesquita Roehl pela amizade, orientação e oportunidade de aprender e crescer mais. Pelo desafio sempre constante para me esforçar ainda mais.

À minha namorada Fernanda Lins Gonçalves Pereira, pelo apoio dado durante a realização deste trabalho, pela amizade, compreensão e paciência, e por todo o caminho que percorremos e percorreremos juntos.

À minha amiga Monique Cordeiro Rodrigues, pela estimada amizade desde a faculdade. Um pouco mais distantes no Mestrado, porém reunidos novamente no Doutorado.

Ao CNPq, à PUC–Rio e à Petrobras, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Aos meus amigos e colegas da PUC–Rio e UERJ pela amizade e por sempre me mostrarem coisas boas da vida.

E a todos aqueles que não foram citados, mas que com certeza influenciaram meu caminho até aqui.

Resumo

Pinto, André Luiz Ferreira; Roehl, Deane de Mesquita (Orientador). **Algoritmo para geração de arranjos de partículas para utilização no método dos elementos discretos**. Rio de Janeiro, 2009. 77p. Dissertação de Mestrado — Departamento de Engenharia Civil, Pontifícia Universidade Católica do Rio de Janeiro.

O método dos elementos discretos (MED), desenvolvido na década de 70, tem despertado, com o aumento da capacidade de processamento e o desenvolvimento de técnicas de computação de alto desempenho, crescente interesse de diversos pesquisadores como ferramenta de estudo de problemas de engenharia. Um campo de estudo de grande apelo é a modelagem de fenômenos associados a materiais granulares, dentre eles a compactação de pacotes granulares — como por exemplo em pós metálicos na indústria siderúrgica —, a produção de areia e a produção de material de sustentação de fraturas estimuladas hidraulicamente na indústria do petróleo, motivação deste trabalho. A aplicação do método requer em sua primeira etapa a geração da configuração inicial das partículas ou o preenchimento de domínios com as mesmas. Alguns estudos têm se voltado para o desenvolvimento de algoritmos de geração de arranjos densos de partículas. Neste trabalho apresenta-se um algoritmo geométrico de geração de arranjos densos de partículas que correspondam a uma dada curva granulométrica e estejam de acordo com outros parâmetros definidos para o pacote granular. No presente trabalho é dada atenção especial a geração de arranjos bidimensionais de discos para modelar o fenômeno de preenchimento de fraturas em poços estimulados por fraturamento hidráulico. O refluxo desse material com o bombeamento de óleo é uma das principais causas de declínio de produção de petróleo em poços fraturados, além de causar danos ao equipamento. O algoritmo proposto foi implementado em linguagem Java e a otimização dos arranjos segundo a porosidade foi realizada através da aplicação de algoritmos genéticos. Aplicações do algoritmo a alguns arranjos de partículas e ao problema de preenchimento de fraturas são apresentados.

Palavras–chave

Método dos elementos discretos; Otimização; Algoritmo genético; Material com gradação funcional;

Abstract

Pinto, André Luiz Ferreira; Roehl, Deane de Mesquita (Advisor). **An algorithm for the generation of particle arrangements for application with the discrete element method**. Rio de Janeiro, 2009. 77p. MSc Dissertation — Departamento de Engenharia Civil, Pontifícia Universidade Católica do Rio de Janeiro.

The Discrete Element Method (DEM), developed in the 70's, has become more a attractive with the increasing computer processing capacity and the development of high-performance computational techniques. This scenario induced growing interest of many researchers as a tool to study engineering problems. A very appealling field of study is the modeling of phenomena associated with granular materials, including the compaction of granular packages, such as metal powders in the steel industry, sand production and proppant flowback in the petroleum industry, which is motivation to this work. The application of the method in its first step requires the generation of the particles' initial configuration or the filling of domains with them. Some studies have focused on the development of algorithms to generate dense packing of particles. This work presents an algorithm to generate random dense packing of particles that correspond to a given granulometric curve and are consistent with other parameters set for the granular package. In the present work special attention is given to generation of two-dimensional packings of disks to model the phenomenon of fractures filling in wells stimulated by hydraulic fracturing. The proppant flowbak generated by the oil pumping is a leading cause of production decline in fractured wells, besides causing damage to the production equipment. The proposed algorithm was implemented in Java language and the optimization of packings was performed according to the porosity using genetic algorithms. Applications of the algorithm to some packings of particles and the problem of filling of fractures are presented.

Keywords

Discrete element method; Optimization; Genetic algorithm; Functionally graded material;

Sumário

Lista de figuras

Lista de tabelas

Lista de abreviaturas

ADD análise de deformação do descontínuo

AE algoritmos evolucionários

AG algoritmos genéticos

GPU Graphics Processing Unit, ou Unidade de Processamento Gráfico

IA inteligência artificial

MEC método dos elementos de contorno

MED método dos elementos discretos

MEF método dos elementos finitos

MGF materiais com gradação funcional

JAXA Japan Aerospace Exploration Agency

JGAP Java Genetic Algorithms Package

JIT compilador *just-in-time*

RBD *random ballistic deposition*

RLP *random loose packing*

1
Introdução

O método dos elementos discretos (MED), como originalmente proposto por Cundall e Strack em 1979 [12], é um método numérico capaz de descrever o comportamento mecânico de um conjunto de discos ou esferas.

Com o aumento da capacidade de processamento e o desenvolvimento de técnicas de computação de alto desempenho, o MED, entre outros, que anteriormente se mantinham em uma esfera mais conceitual ou tinham seu uso limitado a modelos mais simplificados, passaram a despertar crescente interesse de diversos pesquisadores como ferramenta de estudo de problemas de engenharia.

Dentre as aplicações de grande apelo está a simulação de partículas, cada vez mais utilizada para análise de modelos sujeitos a grandes deformações, dentre os quais se destacam os fluidos e materiais granulares. Na área de fluidos, diversos avanços podem ser observados, inclusive com soluções baseada em Graphics Processing Unit, ou Unidade de Processamento Gráfico (GPU), nas área de análise [46] e de animação [34, 43]. Outro campo de estudo é a modelagem de fenômenos associados a materiais granulares dentre eles a compactação de pacotes granulares — como por exemplo em pós metálicos na indústria siderúrgica —, a produção de areia e a produção de material de sustentação de fraturas estimuladas hidraulicamente na indústria do petróleo, motivação deste trabalho.

Hoje em dia o MED pode ser considerado mais adequadamente como uma família de métodos numéricos utilizados para resolver problemas da mecânica aplicada através da discretização do meio empregando partículas. Em problemas onde o material é granular, com ou sem coesão, ou é suscetível à fratura e possível fragmentação, este caráter discreto se mostra interessante. Nesse contexto, em comparação com o método dos elementos finitos (MEF), o MED não precisa recorrer a complexas leis constitutivas [39], sendo assim mais adequado que o primeiro para o tratamento de meios granulares onde as hipóteses clássicas do contínuo são restritivas.

Na formulação original do MED por Cundall e Strack, são permitidas interpenetrações entre as partículas — como na figura 1.1 —, sob condição

de possuírem ordem de grandeza muito pequena em relação ao tamanho das partículas. Nos pontos de contato são geradas forças, normais e tangenciais, proporcionais a tais interpenetrações. Outras forças — como gravidade, força de atrito de Coulomb, etc. — também podem ser computadas e ao final somadas de modo a ser conhecida a força total atuando na partícula. A partir da segunda lei de Newton, calculam-se as acelerações de cada partícula, e as correspondentes velocidades e deslocamentos são obtidas utilizando-se algum método numérico de integração temporal.

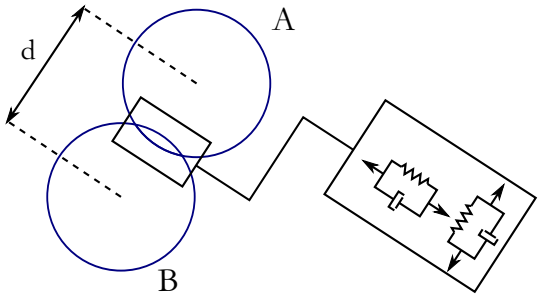

Figura 1.1: Interpenetração entre partículas.

Não obstante inicialmente somente partículas como círculos e esferas terem sido contempladas, atualmente partículas com formatos mais gerais, como poliedros [49], têm sido consideradas, dado que são mais aptas a capturar alguns aspectos essenciais do comportamento mecânico de materiais granulares [30], como atrito entre partículas e embricamento. Este formato angular, no entanto, gera um aumento no trabalho computacional, devido ao aumento da complexidade da detecção de colisão entre as partículas.

O MED como família abrange os seguintes métodos:

- método clássico, originalmente aplicado problemas mecânica das rochas, no qual as deformações de cada partícula individual podem ser desprezadas em comparação às do conjunto;
- o método proposto por Williams et al. [48], que mostra que o MED pode ser visto como um MEF generalizado;
- método de análise de deformação do descontínuo (ADD) proposto por Shi em 1988 [41]. Munjiza [32] afirma que este método é mais adequado a problemas estáticos;
- método combinado de elementos finitos e elementos discretos, como proposto por Munjiza et al. em 1992 [33].

Em comparação com o MEF e com o método dos elementos de contorno (MEC), que já possuem um status mais consolidado, o MED ainda está em uma fase mais preliminar, com muitos campos em franco desenvolvimento. Contudo, podemos observar o emprego do método em algumas áreas, como por exemplo:

- fragmentação de rochas através de explosões [39];
- análise de dutos enterrados [10];
- estabilidade de taludes, figura 1.2 [35];
- impacto de mísseis, figura 1.3 [42];
- corridas de detritos [3];
- simulação de fratura de ossos [29].

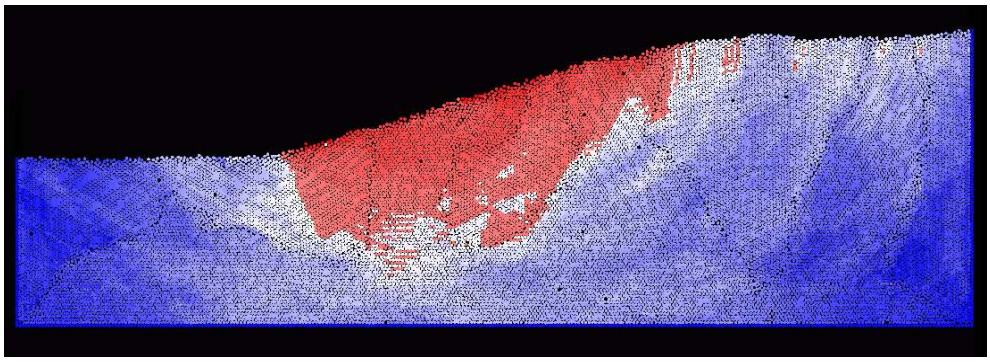

Figura 1.2: Exemplo de análise de estabilidade de taludes.

Ademais do ponto levantado anteriormente sobre o aumento do esforço computacional quando do uso de partículas de geometrias diversas, outro problema diz respeito à quantidade máxima de partículas em uma dada simulação. Embora atualmente simulações com o MED possam conter milhões de partículas, este número ainda não é suficiente para reproduzir completamente algumas situações reais típicas. Usualmente tais simulações restringem o tempo de simulação ou o número de partículas. Presentemente, tem sido explorado o emprego de processamento em paralelo para tentar resolver tais problemas, como pode ser visto em Hustrulid [22].

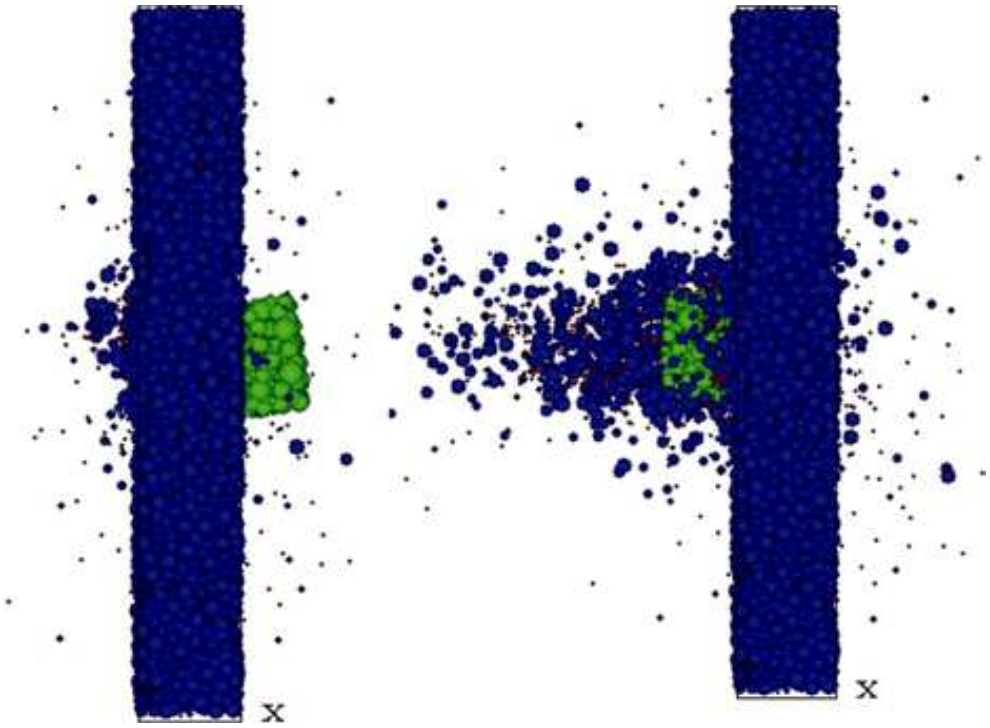

Figura 1.3: Exemplo de simulação de impacto de mísseis.

1.1 Motivação

O primeiro passo em uma simulação do MED é a geração inicial de arranjos de partículas. Hoje em dia, vêm sendo estudados algoritmos de geração de arranjos densos de partículas. Os algoritmos existentes apresentam vantagens e desvantagens. Muitos desses possuem alto custo computacional, o que limita sua utilização. Outros não permitem que a distribuição granulométrica seja prescrita. Alguns geram arranjos com sobreposições das partículas, que produzem, quando do início da análise do MED, forças iniciais nas partículas, não coerentes com problemas nos quais as partículas estão em repouso. Tais algoritmos serão melhor abordados mais a frente.

1.2 Objetivos

O objetivo inicial deste projeto é o desenvolvimento de um algoritmo para a obtenção de um arranjo granular denso que obedeça a uma dada granulometria e que possa ser utilizado como configuração inicial para o cálculo do MED. Não se pode deixar de ter como intenção que tal algoritmo seja capaz de gerar arranjos de milhões de partículas visando seu uso em simulações correntes.

Além do objetivo inicial, outro objetivo considerado interessante é a

otimização do arranjo segundo parâmetros, de forma a tornar possível, por exemplo, prescrever a porosidade.

1.3 Estrutura do trabalho

O segundo capítulo apresenta algoritmos de geração de arranjos desenvolvidos até o presente, classificando-os segundo o tipo de geração e descrevendo seus prós e contras. O terceiro capítulo apresenta o algoritmo proposto e técnicas para sua otimização através de uma estrutura de dados do tipo *quadtree*. Arranjos obtidos com essa implementação são comparados com os gerados pelo programa comercial PFC2D. O quarto capítulo apresenta a otimização de parâmetros dos arranjos gerados utilizando algoritmos genéticos (AG). Finalmente são apresentadas as conclusões dessa pesquisa e propostas para trabalhos futuros no quinto capítulo.

2
Algoritmos de geração de arranjos

O primeiro passo em uma simulação do MED é a geração inicial de arranjos de partículas. Embora uma variedade de algoritmos tenham sido desenvolvidos, não há um consenso sobre uma metodologia ótima a ser empregada na geração dos arranjos.

Os algoritmos desenvolvidos até o presente podem ser classificados em grupos segundo o tipo de geração como: dinâmicos, quasi-estáticos e geométricos. A seguir, serão apresentados os prós e contras de cada grupo e alguns desses algoritmos.

2.1
Algoritmos dinâmicos

Como primeiro algoritmo de geração de arranjos, é natural utilizar o próprio MED para obtenção do arranjo inicial. Tais algoritmos — ditos dinâmicos — têm como vantagem a simplicidade, pois além de já estarem implementados pelos programas de MED, podem ser facilmente aplicados tanto a duas quanto a três dimensões.

Uma abordagem típica é, em primeira instância, a utilização do método *random loose packing* (RLP), que consiste na geração aleatória de um tamanho e uma posição para uma partícula dentro do domínio. Se a partícula não puder ser colocada naquela posição — porque estaria sobreposta a pelo menos uma outra partícula — o tamanho é mantido e é gerada uma nova posição. Este processo é repetido inúmeras vezes até que seja encontrada uma posição aceitável. Entretanto, como pode ser presumido, à medida que são colocadas partículas, torna-se cada vez mais provável que haja uma sobreposição, e portanto uma rejeição, resultando em um domínio com relativamente baixa densidade de partículas ao final da geração. A obtenção de um arranjo mais denso é alcançada através de um processo dinâmico, com a utilização do MED. A aplicação de forças no sistema através da translação das paredes comprimindo o domínio [45] ou da expansão gradual do diâmetro de todas as partículas através do algoritmo Lubachevsky-Stillinger [28] ou de uma versão modificada deste [27]. Esta é a abordagem utilizada no programa PFC2D [1].

Embora estes algoritmos tenham a capacidade de controlar a densidade de partículas do arranjo, tem como desvantagens o elevado tempo computacional necessário para a detecção da sobreposição das partículas e para a simulação do MED.

Outros algoritmos compõem o arranjo através da simulação de uma deposição gravitacional. Como exemplo, temos o método *random ballistic deposition* (RBD) [26, 8], no qual as partículas caem sequencialmente sobre o domínio a partir de posições aleatórias, assentando-se sobre o próprio domínio ou sobre partículas anteriormente assentadas, como na figura 2.1, retirada de Bratberg et al. [8]. Outro exemplo é o algoritmo *hopper to mix* proposto por Feng et al. [18], onde um arranjo regular é colocado sobre um tipo de funil (em inglês hopper) disposto acima do domínio. As partículas são liberadas para passar pelo bico do funil — utilizando o MED sob uma força gravitacional — e neste processo, o arranjo é misturado e assentado densamente no domínio, como visto na figura 2.2, retirada de Feng et al. [18]. Os métodos gravitacionais, além de também sofrerem com o alto esforço computacional inerente ao MED, têm dificuldades em obter arranjos homogêneos, além de não possuírem meios de controlar a densidade do arranjo [37].

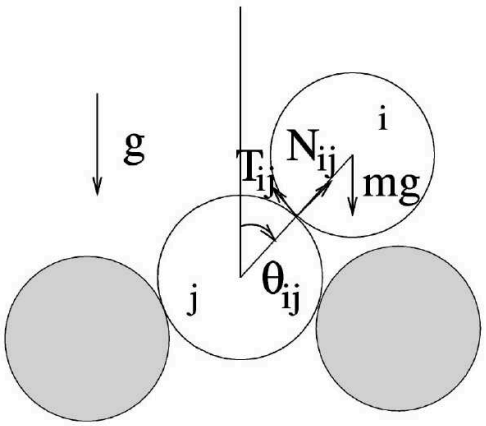

Figura 2.1: Contato durante assentamento segundo o método RBD.

2.2 Algoritmos quasi-estáticos

Os algoritmos quasi-estáticos utilizam simulações de Monte-Carlo para a geração da posição das partículas, ao invés do simulações de MED. Como exemplo temos o algoritmo desenvolvido por Jodrey e Tory [25], como visto na figura 2.3, retirada do mesmo trabalho. Neste algoritmo, inicialmente são gerados aleatoriamente pontos correspondentes aos centros de esferas no domínio. Cada centro tem associada uma esfera interna — que define a

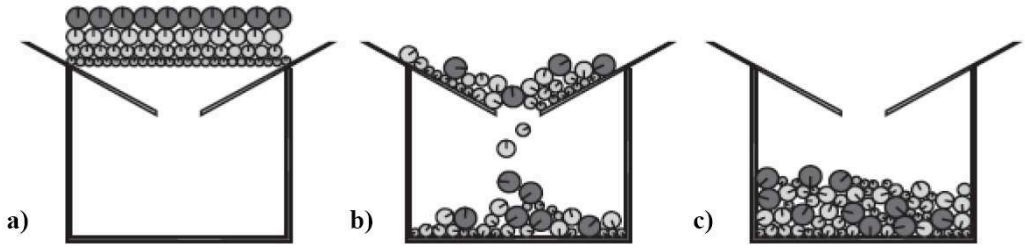

Figura 2.2: Método *hopper to mix*. a) Arranjo regular inicial; b) Simulação do MED; c) Arranjo denso misturado.

densidade verdadeira — e uma externa — que define uma nominal. O algoritmo então passa para um processo de eliminação das sobreposições, enquanto gradativamente reduz o diâmetro externo. O processo termina quando os dois diâmetros se aproximam, e portanto as densidades verdadeira e nominal se igualam. Estes algoritmos podem controlar a densidade de partículas do arranjo e podem gerar arranjos em duas ou três dimensões, contudo a eliminação das sobreposições demanda um grande trabalho computacional.

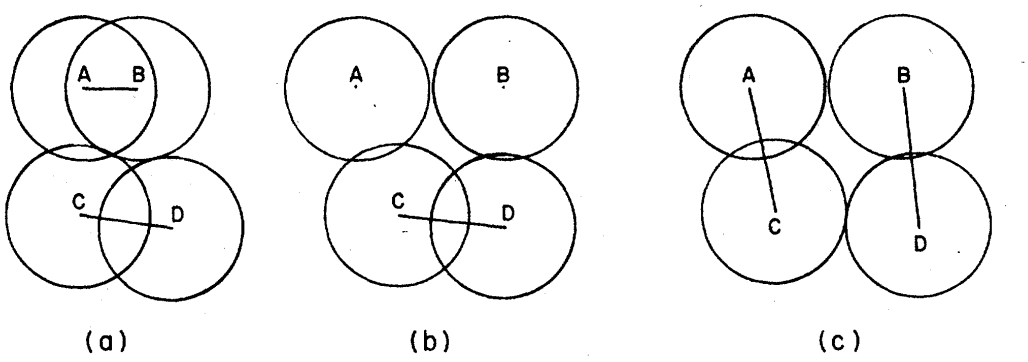

Figura 2.3: Algoritmo de Jodrey e Tory aplicado a um sistema de 4 discos. O processo se dá por eliminação sequencial das maiores sobreposições. a) Eliminação da sobreposição AB; b) Eliminação da sobreposição CD; c) Ao ser eliminada a sobreposição CD, são alistadas as sobreposições AC e BD.

2.3 Algoritmos geométricos

Uma alternativa que pode ser vantajosa aos algoritmos dinâmicos e quasi-estáticos são os algoritmos geométricos. Como o próprio nome diz, utilizam somente cálculos geométricos para a geração de arranjos e, dado que não simulam movimentos das partículas, permitem uma redução substancial do tempo de processamento. Em Feng et al. [18] é proposta uma abordagem chamada *advancing front*. Para esta abordagem, dependendo se o limite do

domínio é incluído no front inicial e como este é formado, duas implementações foram propostas. No algoritmo *open form advancing front* [18], o arranjo é construído camada por camada, como visto na figura 2.4, retirada do mesmo trabalho. Já no *closed form advancing front* [18], o arranjo é inicialmente composto de três partículas dispostas em formato triangular, no meio do domínio. As partículas são, em seguida, adicionadas de modo a tangenciar duas partículas pré-existentes — tomando forma espiral para fora —, como na figura 2.5, retirada de Bagi [7].

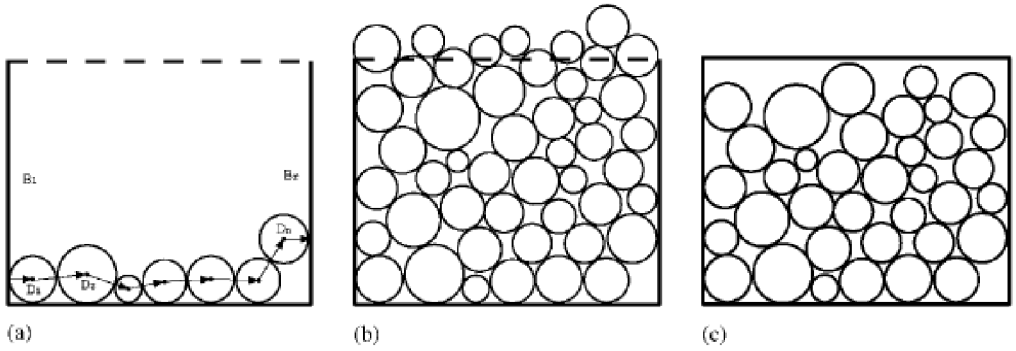

Figura 2.4: Algoritmo *open form advancing front*.

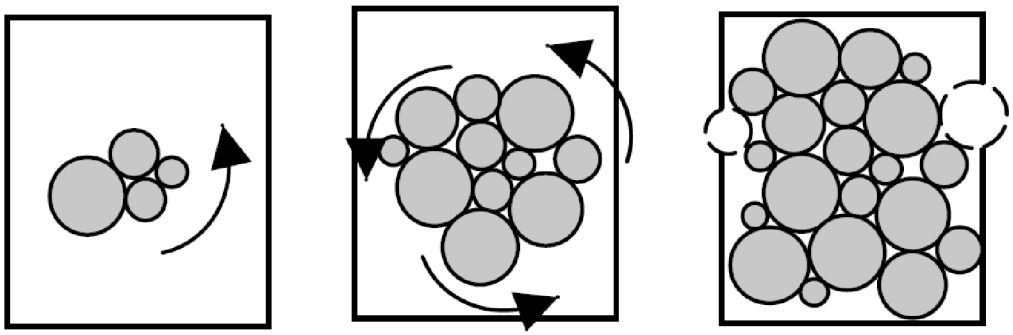

Figura 2.5: Algoritmo *closed form advancing front*.

Uma abordagem similar é proposta por Bagi [7]. Denominada *inwards spiral method*, difere da *closed form advancing front* no que as partículas são colocadas no domínio na forma de uma espiral para dentro, como mostrado na figura 2.6, retirada do mesmo trabalho.

Esses algoritmos, além de precisarem de pouco tempo computacional, permitem que seja obedecida um dada granulometria. Contudo, eles apresentam algumas desvantagens como: a densidade do arranjo não poder ser prescrita e sua difícil implementação no caso tridimensional. Nesse caso, a densidade do arranjo gerado é baixa em relação aos algoritmos dinâmicos.

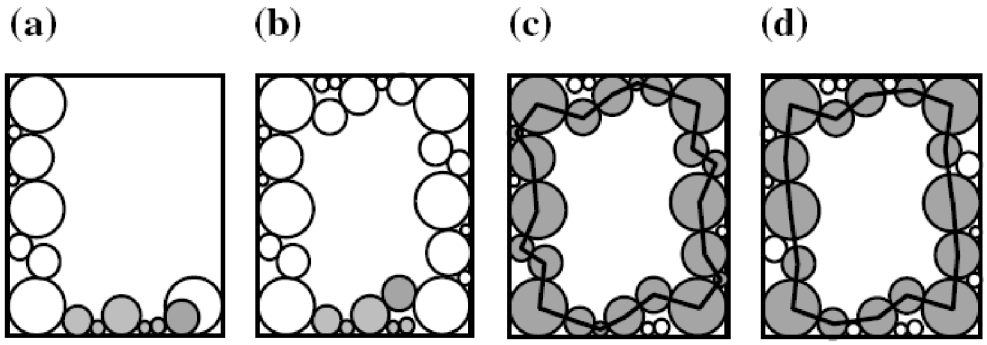

Figura 2.6: Algoritmo *inwards spiral method*.

Nurkanov et al. [36] desenvolveram um algoritmo em três dimensões que tem capacidade de gerar arranjos com densidades compatíveis àquelas encontradas empregando-se algoritmos dinâmicos. Este algoritmo, utilizado no programa S3D SpheroPack [24], opera como se as esferas estivessem sendo despejadas em um recipiente: o tamanho da esfera é escolhido segundo uma dada distribuição e a esfera é "jogada" no recipiente, que é formado pela interseção de N semi-espaços. Tão logo a esfera encontra um obstáculo — parede do recipiente ou outra esfera — ela se adere a ele e desliza na direção da menor energia potencial até encontrar o próximo obstáculo. A esfera pára quando estiver sendo suportada por três superfícies ou por um plano perpendicular à direção de sua queda livre.

Algoritmos capazes de gerar arranjos tanto em duas quanto em três dimensões facilmente são aqueles baseados em triangulação, como os propostos por Cui e O'Sullivan em [11] ou por Jerier et al. em [24]. No algoritmo proposto por Cui e O'Sullivan, primeiramente, é gerada uma quantidade de pontos aleatórios uniformemente distribuídos no domínio. Estes pontos são triangulados, criando uma malha de triângulos em duas dimensões, ou tetraedros em três dimensões. A partir da malha obtida pela triangulação, são gerados círculos inscritos em cada triângulo ou, em três dimensões, esferas inscritas nos tetraedros. De modo a densificar mais o arranjo, o próximo passo é a geração, para cada vértice do triângulo/tetraedro, do maior círculo/esfera que puder ser colocada de modo a tangenciar pelo menos um círculo/esfera anteriormente criados, como visto na figura 2.7 (retirada de Cui e O'Sullivan [11]) para o caso em duas dimensões. Como outros algoritmos geométricos, este algoritmo também tem um custo computacional baixo, embora não possa gerar arranjos com uma dada granulometria. Mais ainda, tais arranjos costumam ter uma densidade baixa. Contudo, deve ser mencionado que a densidade do arranjo é intimamente ligada à malha gerada na triangulação

— principalmente no caso de duas dimensões — e, por conseguinte, visando uma maior densidade, é possível tirar vantagem da melhoria na geração de malhas triangulares/tetraédricas que é utilizada no MEF.

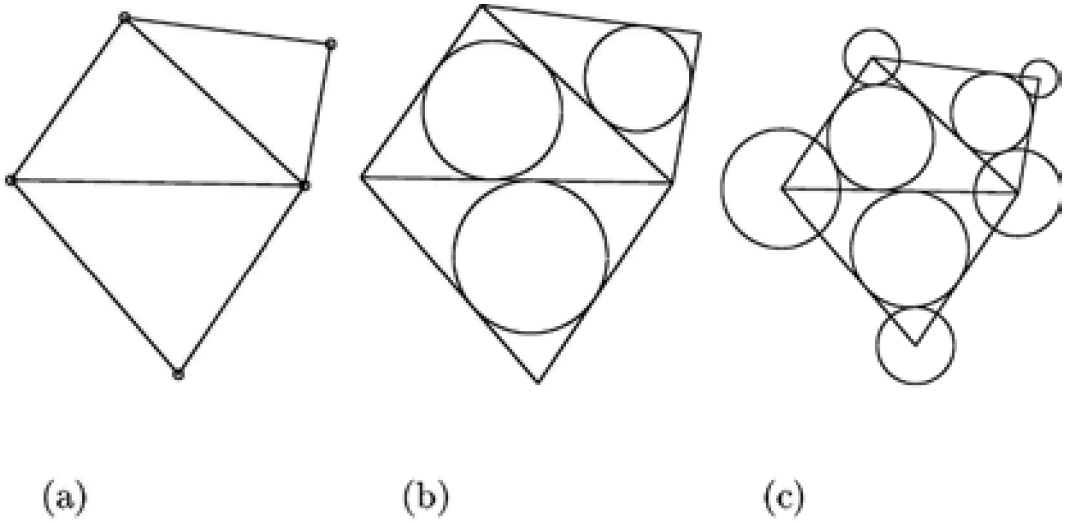

Figura 2.7: Geração por triangulação, como proposto por Cui e O'Sullivan. a) Triangulação dos pontos; b) Geração dos círculos inscritos; c) Geração dos círculos nos vértices.

3
Algoritmo proposto para a geração de arranjos

Neste trabalho, é proposto um algoritmo que busca superar algumas dificuldades encontradas pelos algoritmos anteriormente apresentados. Algumas vantagens deste algoritmo são:

- o algoritmo proposto se enquadra no grupo dos algoritmos geométricos, e portanto, tem um baixo custo computacional na geração de arranjos;
- a distribuição granulométrica é definida pelo usuário. As partículas são geradas aleatoriamente seguindo a granulometria prescrita;
- permite geração de um arranjo de partículas denso;
- como as partículas não se sobrepõem, tem-se que, quando do início da análise do MED, as forças iniciais nas partículas serão zero.

O princípio básico do algoritmo é gerar partículas, uma a uma, de modo que elas sejam posicionadas tangencialmente a outros obstáculos — paredes do domínio ou outras partículas anteriormente colocadas — através da busca de interseção de formas geométricas. Um passo-a-passo detalhado do algoritmo será visto a seguir.

O primero passo do algoritmo proposto é a representação do contorno do domínio. Na implementação neste trabalho, o contorno do domínio é representado através da discretização com círculos, como na figura 3.1. Isto não é uma limitação do algoritmo e o contorno pode ser modelado através de retas ou mesmo linhas curvas. Esta escolha se baseia na metodologia adotada no trabalho de Vieira [47] na implementação do MED, para o qual os modelos gerados neste trabalho serão analisados, onde tanto o contorno quanto as partículas são circulares.

O próximo passo é a criação da lista inicial com as possíveis frentes (em inglês, fronts). A frente será melhor abordada mais adiante — quando da sua utilização no algoritmo —, porém, para fins de continuidade, neste passo esta lista é gerada simplesmente ordenando-se as partículas constituintes do contorno segundo uma dada metodologia, de modo a ser garantida uma hierarquia definida. Esta metodologia é arbitrária e foi escolhida de maneira a priorizar as partículas posicionadas mais abaixo e em seguida, como critério

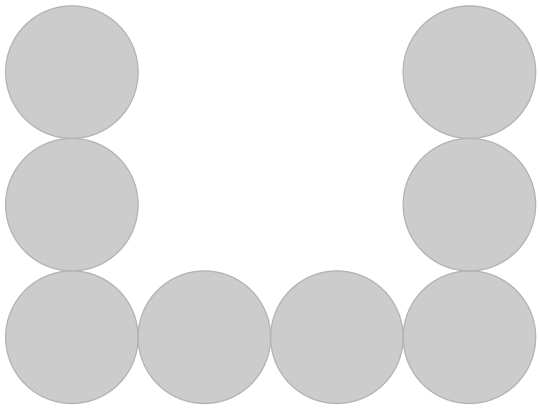

Figura 3.1: Representação do contorno por círculos.

de desempate, aquelas mais à esquerda. Como duas partículas não podem ter um mesmo par de coordenadas, estes critérios são necessários e suficientes para identificar univocamente qualquer partícula em uma análise em duas dimensões.

Em seguida, inicia-se um laço (*loop*) de geração de partículas. Para ser gerada uma partícula, deve-se determinar o tamanho da partícula, além de ter conhecimento das fronteiras do domínio e das partículas anteriormente geradas, de modo a ser possível a geração sem sobreposição. No caso implementado, como as partículas são circulares, o tamanho corresponde ao diâmetro.

Para a geração de cada partícula, inicialmente, obtém-se na lista de possíveis frentes a primeira destas (figura 3.2). Como descrito anteriormente, esta frente nada mais é que uma partícula, e em uma primeira iteração é uma partícula discretizada do contorno. A idéia por trás do algoritmo é gerar partículas que estejam em contato com a frente e com outro obstáculo — a partir deste ponto, como o contorno do domínio é representado por partículas, não será mais chamado obstáculo e sim, partícula.

Depois, para esta frente, procuram-se todas as partículas localizadas em suas proximidades, e por conseqüência, que possam interferir na posição da partícula sendo gerada. A definição de proximidade é relativa, e portanto, neste caso foram arbitradas como estando próximas, teoricamente, todas as partículas que estivessem situadas dentro da área de busca representada por um quadrado com centróide coincidente ao da frente e lado igual a duas vezes a soma do raio da partícula frente, do diâmetro da partícula que está sendo gerada e do raio da maior partícula possível, conforme visto na figura 3.3.

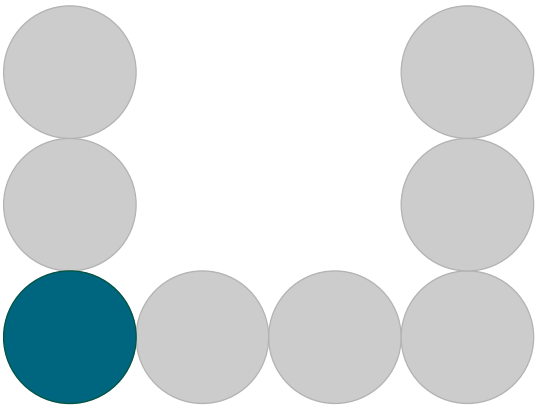

Figura 3.2: Primeira frente — uma das partícula mais abaixo e aquela mais à esquerda.

Na prática, no entanto, um cuidado extra deve ser tomado — o quadrado que descreve a área de busca de proximidade de partículas tem suas bordas superior e direita como intervalo aberto, e por isso, no caso de uma partícula estar localizada sobre uma destas bordas, ela não seria computada para cálculo de interseções. Para resolver tal situação, o lado do quadrado deve ser necessariamente maior do que o calculado anteriormente, porém, quanto maior a área de busca, mais partículas seriam consideradas próximas, aumentando o esforço computacional desnecessariamente. No escopo da implementação deste algoritmo foi arbitrado que ao invés de ser somado o diâmetro da partícula a ser gerado, soma-se este valor incrementado de dez porcento.

Com todas as partículas próximas em mãos, para cada uma delas, assim como para a frente, geram-se o que foram chamados halos (em inglês, também, halos), mostrados na figura 3.4. Halo é um recurso que será utilizado para o cálculo do posicionamento possível do centróide da partícula, e pode ser definido como o lugar geométrico dos pontos cuja distância até qualquer ponto na superfície da partícula é maior ou igual à metade do tamanho da partícula a ser gerada. No caso específico de um círculo de diâmetro d_1, o halo é uma circunferência de diâmetro $d_1 + d_{max}$, onde d_{max} é o tamanho máximo das partículas.

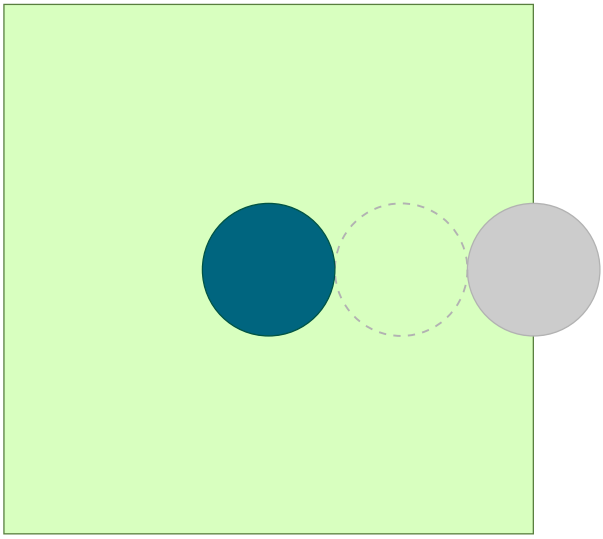

Figura 3.3: Área de busca. A área de busca teórica é representada pelo quadrado verde. O círculo azul representa a frente, o cinza uma partícula com o tamanho máximo e o tracejado seria uma possível partícula a ser gerada. Nota-se que a área de busca engloba toda a área onde pode haver uma partícula que influencie a geração.

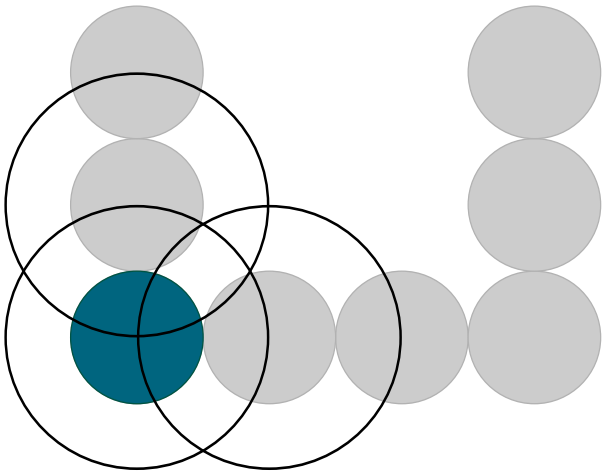

Figura 3.4: Halos das partículas próximas à primeira frente.

O uso do halo se justifica por definir todas as posições candidatas a centróide da partícula a ser gerada. Estas posições determinam-se pelas interseções do halo da frente com cada halo das partículas próximas (figura 3.5). Algumas destas posições, todavia, são impraticáveis, dado que podem estar situadas fora do domínio ou no interior de outros halos (figura 3.6). Em tais circunstâncias, estas posições são descartadas, restando somente as que

foram classificadas no algoritmo como posições praticáveis (figura 3.10).

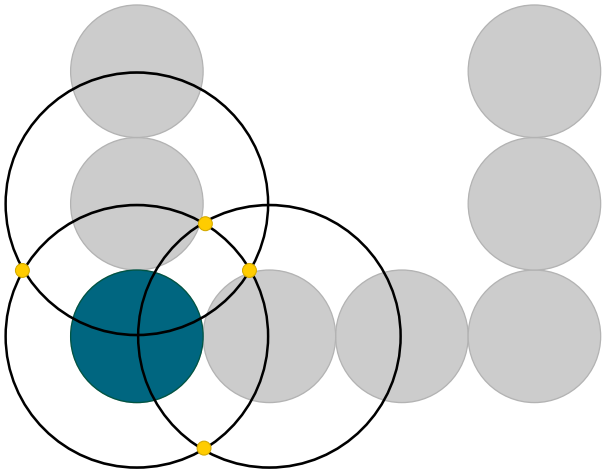

Figura 3.5: Posições candidatas a ser centróide da partícula sendo gerada para a primeira frente.

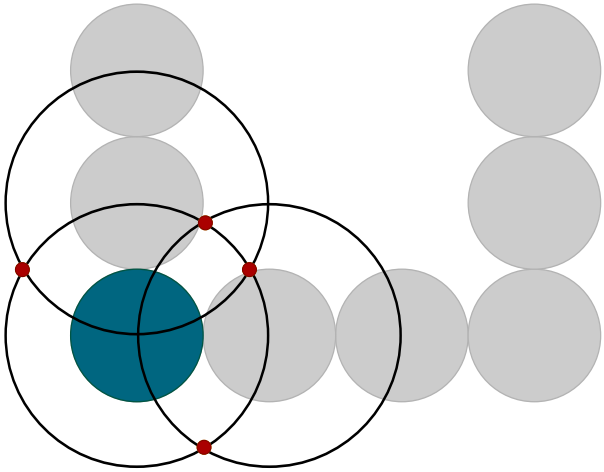

Figura 3.6: Descarte das posições impraticáveis do centróide da partícula a ser gerada para a primeira frente. Não há boas interseções para essa frente.

Há a possibilidade, no entanto, que não se tenha conseguido, para a frente selecionada, encontrar uma interseção sequer onde pudesse ser gerada uma partícula. Neste caso, a frente é descartada da lista de possíveis frentes. Para fins de otimização, se não houver na lista de boas interseções pelo menos duas delas, a frente também é removida de sua lista. A próxima iteração é

iniciada conservando-se, entretanto, o tamanho anterior da partícula (figuras 3.7, 3.8, 3.9 e 3.10).

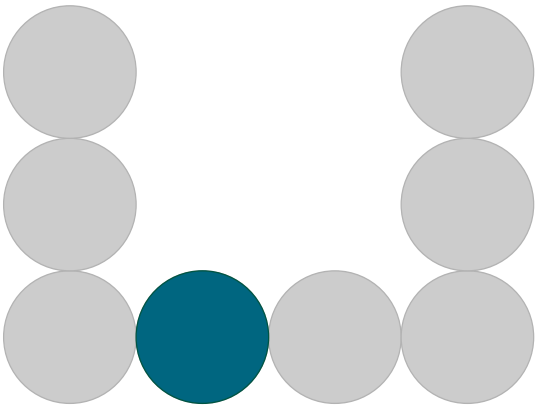

Figura 3.7: Segunda frente — uma das partícula mais abaixo e a segunda mais à esquerda.

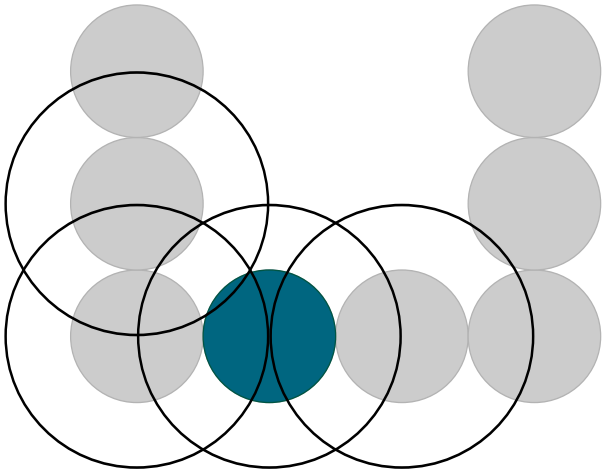

Figura 3.8: Halos das partículas próximas à segunda frente.

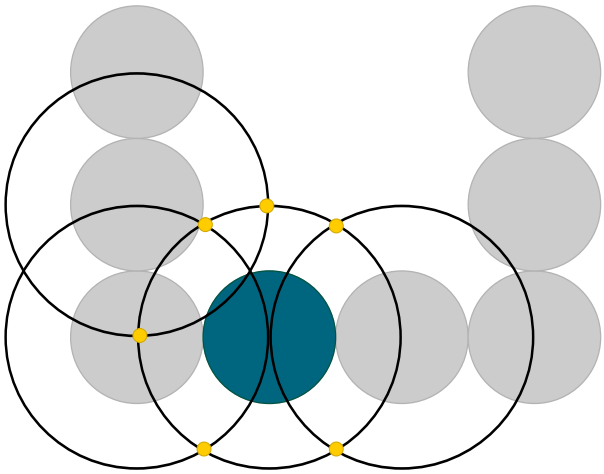

Figura 3.9: Posições candidatas a ser centróide da partícula sendo gerada para a segunda frente.

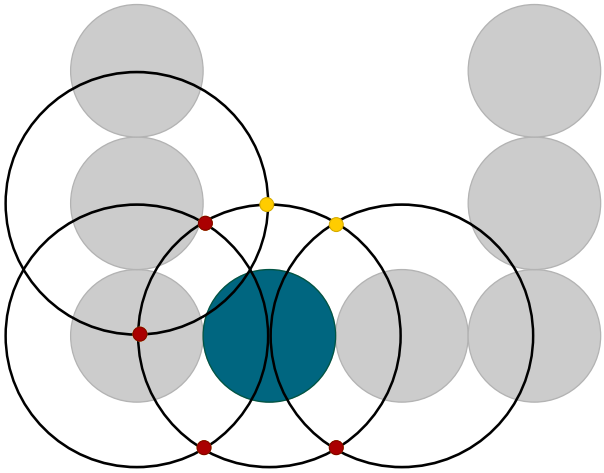

Figura 3.10: Descarte das posições impraticáveis do centróide da partícula a ser gerada para a segunda frente.

Dentre as boas interseções, somente uma pode ser escolhida — e ser designada no algoritmo como melhor interseção (figura 3.11) — e para isso é utilizado o mesmo critério que antes, onde a vantagem é dada à interseção que possua a menor ordenada. Este critério tende a levar a partícula a uma posição de energia potencial mais reduzida, na tentativa de simular mais fidedignamente uma situação de deposição da partícula.

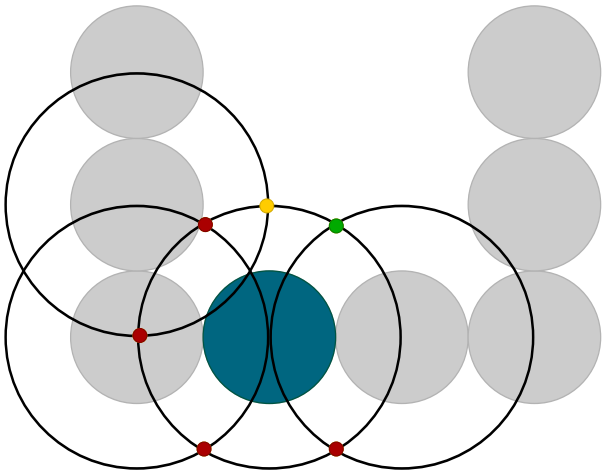

Figura 3.11: Escolha da melhor entre as posições praticáveis do centróide da partícula a ser gerada para a segunda frente.

Ao final, no caso favorável de ser encontrada uma interseção, é gerada uma partícula com o tamanho selecionado e centróide nesta interseção, como apresentado na figura 3.12. A partícula é então colocada no topo da lista das possíveis frentes, e tem começo uma nova iteração, como na figura 3.13.

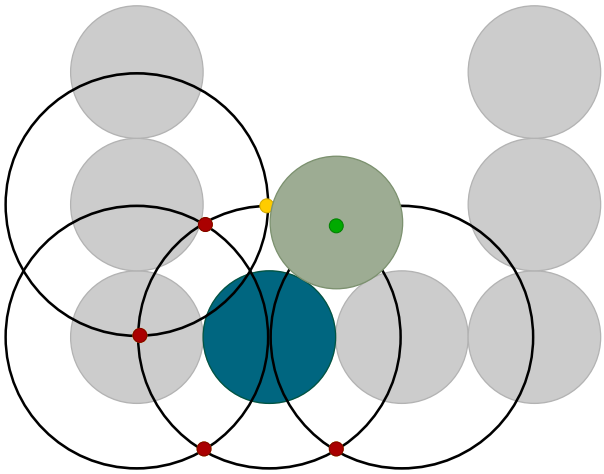

Figura 3.12: Geração de nova partícula na melhor posição do centróide para a segunda frente.

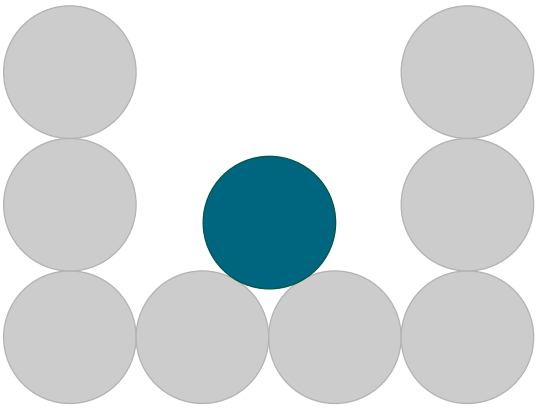

Figura 3.13: Inserção da nova partícula no topo da lista de frentes.

As iterações terminam quando se chega ao final da lista de possíveis frentes ou a lista de tamanhos de partículas. A lista de tamanhos de partículas é definida no início do algoritmo e é constituída por um número suficiente de valores de tamanhos que permitem que o domínio seja preenchido completamente. Para se obter esta lista, é necessário estimar a quantidade máxima de partículas. Esta estimativa foi avaliada com base no domínio e no menor tamanho das partículas, segundo a fórmula 3-1.

$$n_{est} = \left\lceil \frac{a_c}{a_{p,min}} \right\rceil \tag{3-1}$$

Onde,

n_{est} quantidade estimada de partículas

a_c área limitada pelo domínio

$a_{p,min}$ área da partícula mínima

3.1 Otimização do desempenho do algoritmo

Uma preocupação com relação ao algoritmo é que a busca das partículas próximas à frente pode ser extremamente custosa computacionalmente, principalmente tendo-se em vista que será gerado um número considerável de partículas e somente poucas delas podem estar efetivamente em contato com a frente. Para avaliar o desempenho relativo desta busca, foi executado um

benchmark na implementação do algoritmo, de maneira que pudesse ser confirmado que o gargalo da aplicação seria a busca. O domínio é quadrado de lado 100mm e a granulometria é definida pela tabela 3.1. A tabela 3.2 apresenta o resultado do *benchmark*, mostrando somente os dez métodos mais custosos computacionalmente, em porcentagem.

Tabela 3.1: Granulometria utilizada no *benchmark*.

Diâmetro(mm)	Passante acumulado(%)
1,275	0
1,800	35
2,550	95
3,540	100

Tabela 3.2: *Benchmark* da implementação inicial do algoritmo.

Método	Tempo		Invocações
Todos	49.796 ms	(100,0%)	
ListNode.getPoints()	16.693 ms	(33,5%)	10.317
LeafNode.getPoints()	5.403 ms	(10,9%)	10.915.714
query(Rectangle2D)	1.847 ms	(3,7%)	10.315
containsPoint(Particle, Point2D)	1.132 ms	(2,3%)	1.142.197
getGoodIntersections(List, List)	1.017 ms	(2,0%)	5.158
containsPoint(CircularParticle, Point2D)	733 ms	(1,5%)	1.142.197
intersections(CircularParticle, ...)	192 ms	(0,4%)	107.078
makeHalo(double)	112 ms	(0,2%)	112.236
intersections(Particle, Particle)	108 ms	(0,2%)	107.078
getIntersections(Particle, List)	101 ms	(0,2%)	5.158

Pode-se notar que os dois primeiros métodos são responsáveis por 44,4% do esforço computacional da implementação do algoritmo e portanto uma ligeira explicação destes se torna necessária, de modo a dar prosseguimento ao texto.

Os método `ListNode.getPoints()` e `LeafNode.getPoints()` são chamados pelo método `rangeSearch(Rectangle2D)`, que por sua vez, é chamado pelo método `getNearParticles(Particle, double)`. Este método é responsável por retornar as partículas que estariam próximas a uma dada partícula e como esperado, o gargalo é a busca das partículas. Uma possível maneira de melhorar o desempenho do algoritmo de busca por partículas próximas é utilizar uma estrutura de dados, como é explicado a seguir.

Estruturas de dados são um modo de armazenar e organizar dados de forma a poderem ser utilizados eficientemente por um computador. Diferentes estruturas de dados são adequadas a diferentes problemas e podem ser ex-

tremamente especializadas. Para problemas de particionamento de um espaço bidimensional, *quadtrees* se mostram uma estrutura de dados interessante [40].

3.1.1 Quadtrees

As *quadtrees* são árvores baseadas na decomposição recursiva do espaço bidimensional em quatro regiões — normalmente chamados quadrantes nordeste, sudeste, sudoeste e noroeste —, daí sua denominação. Seu análogo em três dimensões é a *octree*. Diversos tipos de *quadtree* foram propostas, cada uma adaptada a diferentes tipos de busca em domínios espaciais. Um exemplo natural em um espaço bidimensional é a busca por latitudes e longitudes de cidades, entretanto, qualquer procura em um banco de dados que contenha dois parâmetros também pode ser considerado assim, como uma busca por pessoas entre um certo intervalo de idade e de renda, por exemplo. Por ser possível representar pares de parâmetros como coordenadas de pontos em um gráfico, eles serão referidos como pontos de dado.

Embora os vários tipos de *quadtree* compartilhem algumas características básicas, algumas diferenças podem ser notadas. Segundo Sperber [44], existem alguns tipos básicos de *quadtrees*, que serão brevemente explicadas.

Point quadtree

Point quadtree é uma combinação de grade uniforme e árvore binária proposta por Finkel e Bentley [19] (figura 3.14, retirada de Samet [40]).

O método de construção é colocar o primeiro ponto de dado no nó da raiz e depois subdividir o espaço em quadrantes, tendo o ponto como centro da subdivisão. As inserções subseqüentes são recursivamente aplicadas através da descoberta do quadrante no qual o ponto está contido e, no caso de estar vazio, da adição do ponto e seguinte subdivisão em outros quadrantes, como anteriormente feito. Se, por outro lado, o quadrante não estiver vazio, procura-se o subquadrante onde o ponto está situado.

Do mesmo modo que a árvore binária, a estrutura do *point quadtree* depende da ordem dos pontos inseridos.

Há alguns algoritmos para deleção, mas eles não são considerados ideais.

A *point quadtree* é especialmente indicada para busca de vizinho mais próximo (em inglês, *nearest neighbour search*).

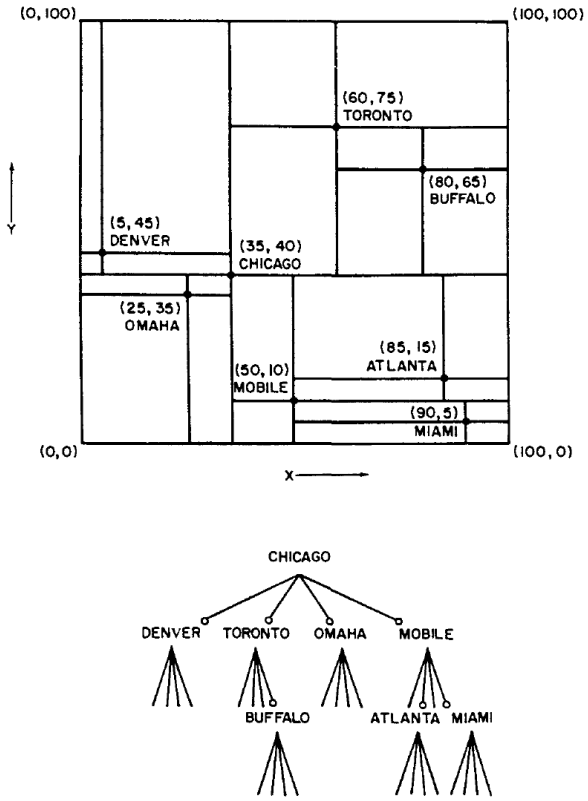

Figura 3.14: *Point quadtree* e os dados que ela representa.

MX quadtree

Outro tipo de *quadtree* são as chamadas *region quadtrees*. As *region quadtrees* têm por característica dividir o espaço regularmente. Há muitas versões de *region quadtree*, uma delas chamada *MX quadtree*. A figura 3.15, modificada de figura retirada de Samet [40], mostra o processo de inclusão de um primeiro ponto na *quadtree*. A figura 3.16, retirada de Samet [40], mostra uma *MX quadtree* com vários pontos incluídos.

A *MX quadtree* decompõe um espaço finito em regiões quadradas de dimensões um por um, como uma matriz — MX vem de *matrix*, em inglês — e seus nós-folha indicam se existe ou não um ponto de dado naquela posição na matriz.

Os pontos na *MX quadtree* são envoltos por um quadrado de lado 2^N, onde N é inteiro, dado que cada subdivisão da região ocorre no ponto central. Se for necessário representar uma região que não seja quadrada, deve ser criada uma *MX quadtree* que seja grande o suficiente para envolver toda a região.

Diferentemente da *point quadtree*, somente os nós-folhas podem armazenar dados.

Essas árvores são bem eficientes em termos de memória no caso de representação de matrizes esparsas. É uma boa opção para busca de k-vizinhos

mais próximos (em inglês, *k-nearest neighbors search*).

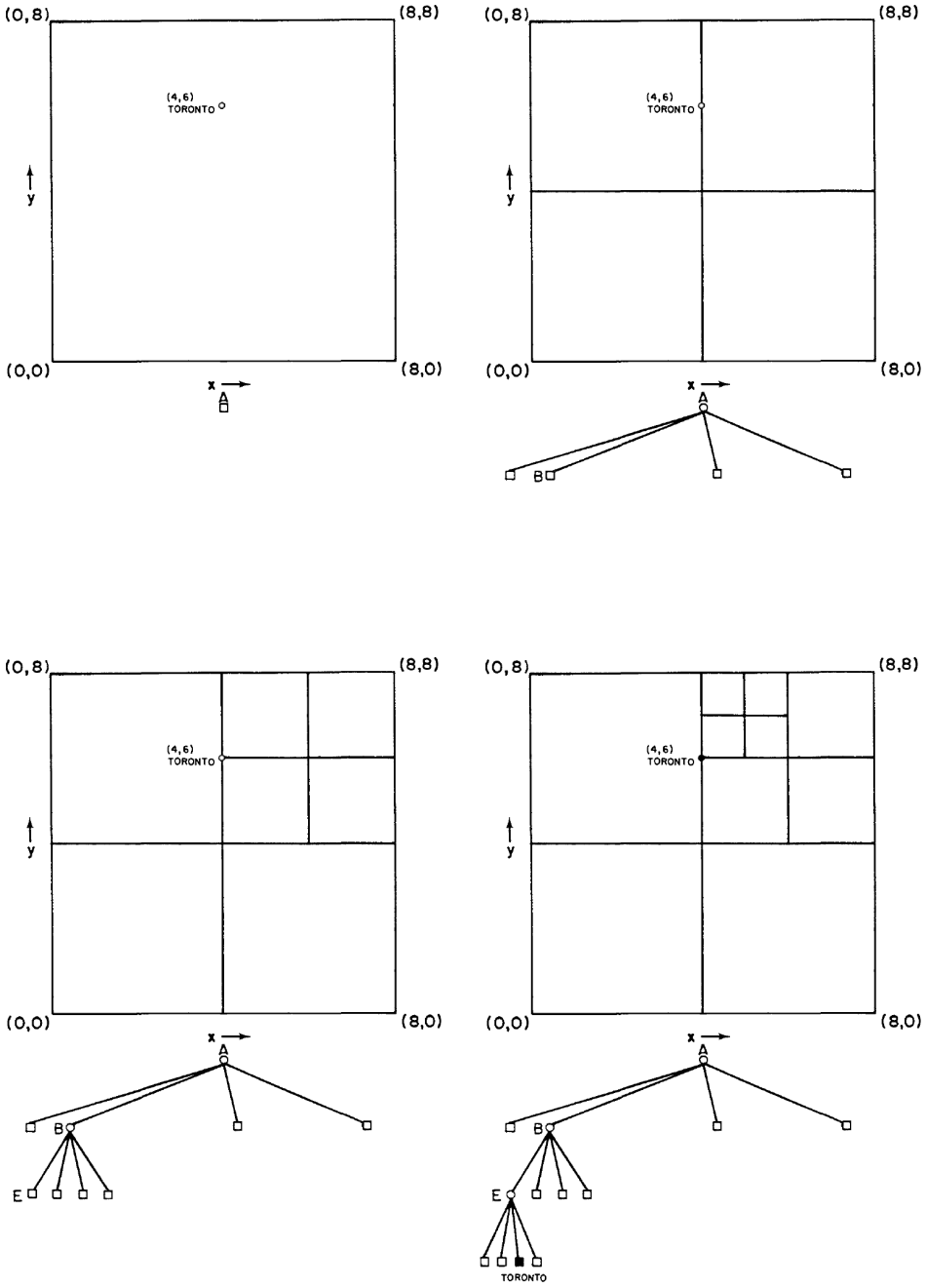

Figura 3.15: Processo de inclusão de um ponto na *MX quadtree*.

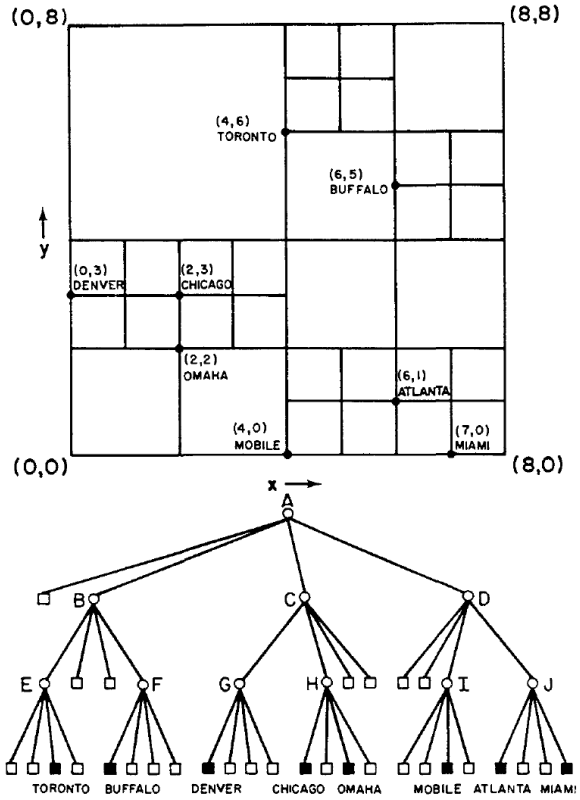

Figura 3.16: *MX quadtree* e os dados que ela representa.

PR quadtree

Outra *region quadtree* é a *PR quadtree* (figura 3.17, retirada de Samet [40]), cuja diferença em relação à *MX quadtree* é que a subdivisão em quadrantes é baseada nos pontos inseridos. Os pontos são inseridos como numa *point quadtree*, mas, como numa *MX quadtree*, os dados são inseridos nos nós-folha — PR abrevia *point region*. Na inserção de pontos de dados, ao ser alcançado um nó-folha que não esteja vazio, o espaço precisa ser subdividido até que somente um ponto esteja contido por quadrante. Isto torna a inserção mais complicada que em uma *point quadtree* ou em uma *MX quadtree*.

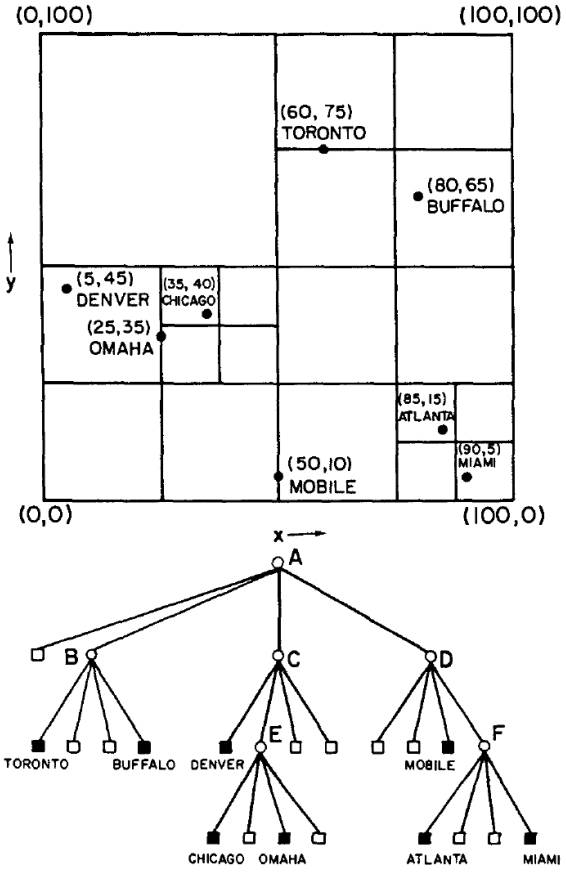

Figura 3.17: *PR quadtree* e os dados que ela representa.

Bucket PR quadtree

A *bucket PR quadtree* é basicamente uma *PR quadtree*. A diferença é que o espaço é subdividido somente quando o número de pontos de dados em cada nó extrapola a capacidade do balde (*bucket*), como visto na figura 3.18 para uma capacidade do balde igual a 2 (retirada de Samet [40]).

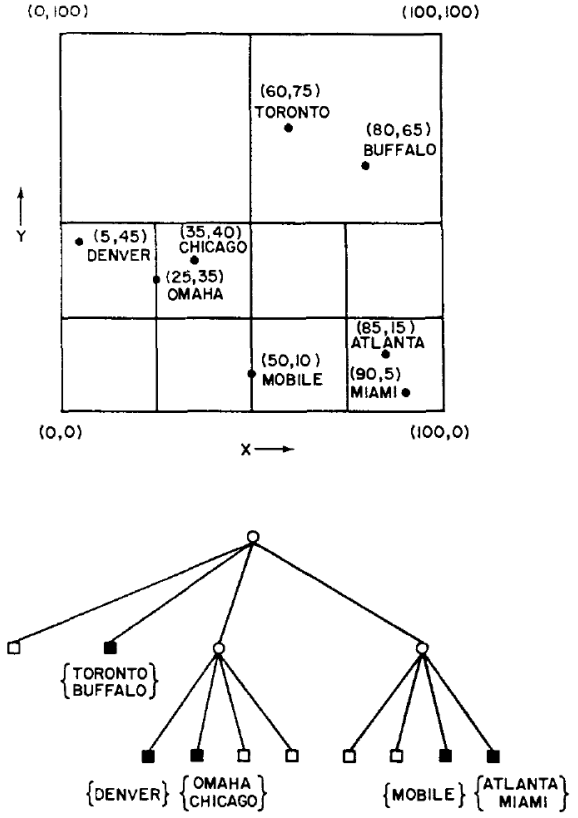

Figura 3.18: *Bucket PR quadtree* para uma capacidade do balde igual a 2 e os dados que ela representa.

3.1.2 Escolha da quadtree

Como o problema abordado na implementação do algoritmo é a busca pelas partículas que estejam próximas à frente e o espaço a ser preenchido é finito, a escolha da *MX quadtree* é natural. Seu desempenho na implementação do algoritmo foi avaliado e os dez métodos mais custosos computacionalmente são mostrados na tabela 3.3, para os mesmos parâmetros da avaliação anterior.

Ao serem comparados os *benchmarks* relativos às duas implementações, a original e aquela que utilizou a *MX quadtree*, notamos uma melhora significativa no desempenho — uma redução de 67,3% no tempo computacional —, e os métodos críticos na avaliação anterior foram mitigados: o método `ListNode.getPoints()`, que possui como contrapartida na segunda implementação o método `QuadtreeNode.getPoints()`, que tomava 16.693ms (33,5%), passou a 185ms (1,1%), enquanto o método `LeafNode.getPoints()`, o segundo mais custoso na implementação original com 10,9% e que tomava 5.403ms, não mais figura sequer entre os dez métodos mais custosos na segunda implementação, com menos de 0,8% do total — para constar, toma 56ms, cerca de 0,3%.

Tabela 3.3: *Benchmark* da implementação do algoritmo utilizando a *MX quadtree*.

Método	Tempo		Invocações
Todos	16.287 ms	(100,0%)	
query(Rectangle2D, Node, Rectangle2D)	1.875 ms	(11,5%)	225.742
containsPoint(Particle, Point2D)	1.380 ms	(8,5%)	1.280.257
getGoodIntersections(List, List)	1.074 ms	(6,6%)	5.128
containsPoint(CircularParticle, Point2D)	727 ms	(4,5%)	1.280.257
getQuadrantBounds(Rectangle2D, Quadrant)	338 ms	(2,1%)	732.169
getSon(Quadrant)	267 ms	(1,6%)	761.917
Quadrant.values()	212 ms	(1,3%)	186.872
intersections(CircularParticle, ...)	205 ms	(1,3%)	128.168
QuadtreeNode.getPoints()	185 ms	(1,1%)	77.360
makeHalo(double)	133 ms	(0,8%)	133.326

Algo que deve ser ponderado na análise dos *benchmarks* é que o tempo registrado não pode ser levado em consideração em termos absolutos, já que há uma sobrecarga computacional gerada pelos próprios *benchmarks*. Para fins de medição relativa de custo computacional entre os métodos, no entanto, esses valores são perfeitamente válidos. Nesse sentido, nota-se claramente que o objetivo da redução do trabalho computacional foi cumprido com a utilização da estrutura de dados escolhida, mostrando-se uma decisão acertada.

3.2 Validação do algoritmo

O processo de validação dos resultados ocorreu em duas partes: primeiramente foi conduzida uma comparação com o programa PFC2D e a seguir, uma série de testes foi realizada para por à prova o desempenho e a escalabilidade do algoritmo.

3.2.1 Comparação com algoritmos utilizados no programa PFC2D

Para efeito de comparação, está sendo reproduzida a geração de arranjo através do programa PFC2D como vista em Huamán [21]. No trabalho supracitado foram estudados quatro cenários de preenchimento de fraturas com partículas de sustentação. Neste trabalho, só irá ser reproduzido o cenário 1, pois esse foi o único a ser modelado em duas dimensões.

O domínio tem por dimensões largura e comprimento normalizado dados, respectivamente, pelas fórmulas $W = (W_r + 2) \cdot d_p$ e $L_r = L/d_p$, onde a largura normalizada W_r é igual a 4,88 e o diâmetro médio d_p, 1,25mm. O comprimento normalizado adotado foi igual a 50, *apud* Asgian et al. [6],

cujo trabalho concluiu, após inúmeras simulações numéricas, que este valor representa adequadamente um modelo para estudo do refluxo de propantes utilizando o MED.

Conseqüentemente, as dimensões finais a serem utilizadas nesta comparação são largura igual a 8,6mm e comprimento igual a 62,5mm.

A granulometria segue a distribuição dada pela tabela 3.4 e é mostrada na figura 3.19.

Tabela 3.4: Dados granulométricos utilizados.

Diâmetro(mm)	Passante acumulado(%)
1,200	0
1,215	11
1,225	21
1,235	31
1,245	43
1,255	54
1,265	65
1,275	77
1,285	89
1,295	100

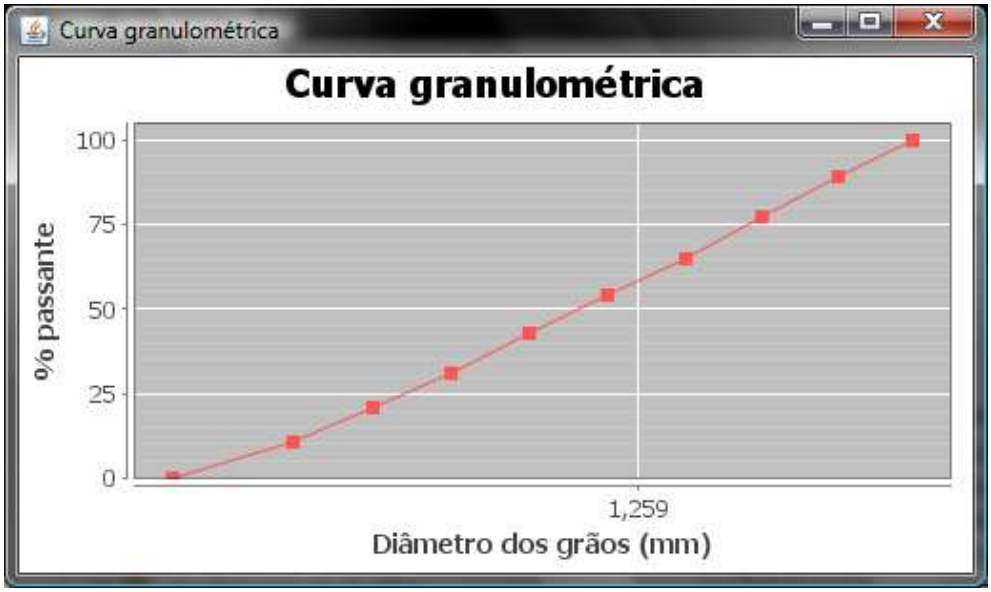

Figura 3.19: Curva granulométrica utilizada.

Para a geração do arranjo através do programa PFC2D, utiliza-se uma linguagem de programação que se encontra embutida nele chamada FISH. Esta linguagem permite ao usuário definir novas variáveis e funções, extendendo assim o programa PFC2D [1].

No PFC2D não existe uma maneira generalizada de geração de arranjos e por isso, alguns métodos serão abordados: a geração por tentativas, por expansão do raio e por repulsão explosiva.

Geração através do algoritmo proposto

Para a geração através do algoritmo proposto, são necessários somente como parâmetros o domínio e a granulometria. Os resultados podem ser vistos na tabela 3.5 e um exemplo é mostrado na figura 3.20.

Tabela 3.5: Resultado da geração através do algoritmo proposto.

Tempo(ms)	Partículas geradas	Porosidade
234	339	0,225
46	338	0,227
47	339	0,224

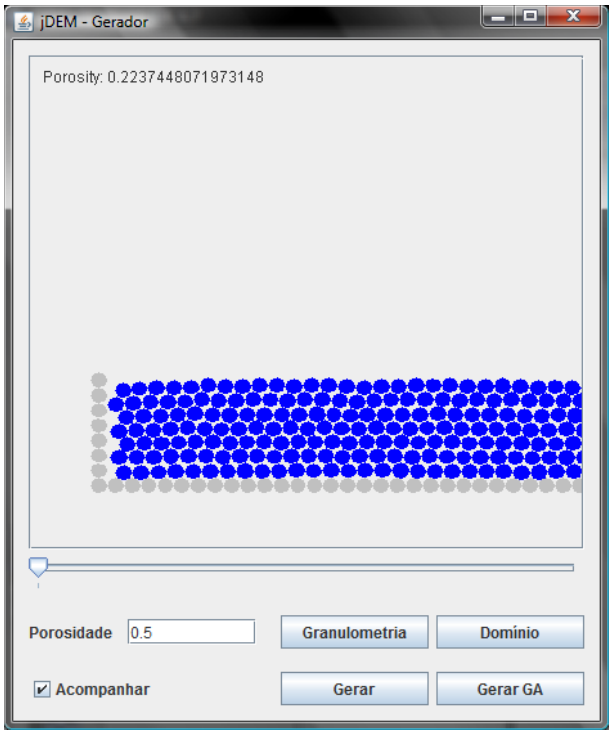

Figura 3.20: Geração através do algoritmo proposto.

Embora o tempo da primeira geração seja muito superior ao das subseqüentes, não há erro nestes valores. Essa discrepância foi produzida pelo compilador *just-in-time* (JIT) do Java.

O JIT converte, em tempo de execução, as instruções de *bytecode* para código de máquina, incrementando o desempenho. O código de máquina resultante da compilação JIT é armazenado na memória, garantindo que o trecho de código em questão não será mais recompilado ou reinterpretado

sempre que, durante a execução do programa, for novamente acionado. Além disso, muitos compiladores JIT possuem mecanismos para realizar otimizações adicionais nos trechos de código do programa que são executados com maior freqüência.

De forma a obter resultados mais homogêneos, antes de cada execução, o programa foi recompilado. Desta forma, simula-se o ato de executar o programa pela primeira vez. Fica óbvio que, no caso de um uso mais freqüente, a velocidade seria aumentada. Estes segundos resultados podem ser vistos na tabela 3.6.

Tabela 3.6: Resultado da geração através do algoritmo proposto, sem otimização.

Tempo(ms)	Partículas geradas	Porosidade
218	339	0,225
218	337	0,224
219	338	0,225

Nos métodos a seguir, é passado como parâmetro a porosidade. Para efeito de comparação, dado que o algoritmo proposto neste trabalho não prescreve uma porosidade, um valor médio obtido do arranjo acima é adotado, igual a $0,225$.

Geração por tentativas no programa PFC2D

O primeiro método de geração testado no PFC2D foi o por tentativas. Como o nome implica, esse método consiste em gerar um arranjo colocando partículas aleatoriamente no domínio, sem que haja sobreposição. Como é possível imaginar, quanto mais cheio estiver o domínio, mais difícil será colocar outra partícula.

Nesse método, são necessários como parâmetros: o domínio, a granulometria e o número máximo de partículas n_{max} que tentarão ser colocadas. Ressalta-se aqui que, por imposição do programa PFC2D, a granulometria foi estipulada uniformemente distribuída entre $1,200$ e $1,295$. Os resultados podem ser vistos na tabela 3.7 para $n_{max} = 20.000$ — que é o padrão no programa — e um exemplo é mostrado na figura 3.21.

Pode-se notar que a porosidade do arranjo é muito elevada, e no intuito de tentar diminuí-la, gerou-se novamente arranjos, só que desta vez, utilizando $n_{max} = 200.000$ com os resultados obtidos mostrados na tabela 3.8, e $n_{max} = 2.000.000$, na tabela 3.9.

Uma comparação destes resultados pode ser vista no gráfico 3.22. Pode-se notar um aumento muito grande no tempo computacional para um ganho

Tabela 3.7: Resultado da geração por tentativas no programa PFC2D — 20.000 tentativas.

Tempo(ms)	Partículas geradas	Porosidade
30	213	0,516
50	210	0,523
30	203	0,537

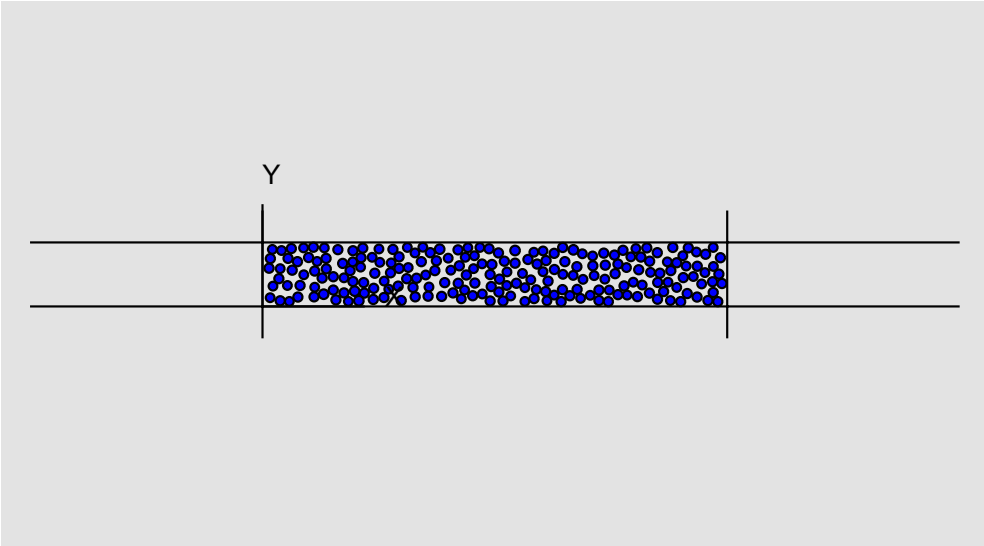

Figura 3.21: Resultado da geração por tentativas no programa PFC2D.

Tabela 3.8: Resultado da geração por tentativas no programa PFC2D — 200.000 tentativas.

Tempo(ms)	Partículas geradas	Porosidade
200	217	0,509
210	221	0,497
200	214	0,513

Tabela 3.9: Resultado da geração por tentativas no programa PFC2D — 2.000.000 tentativas.

Tempo(ms)	Partículas geradas	Porosidade
1820	231	0,474
1810	229	0,479
1820	227	0,482

muito reduzido na porosidade, como era esperado. Este método não é recomendado nem mesmo pelo manual do PFC2D [1], sendo aconselhado escrever uma função em FISH para automatizar o processo que assegure que a porosidade desejada seja obtida .

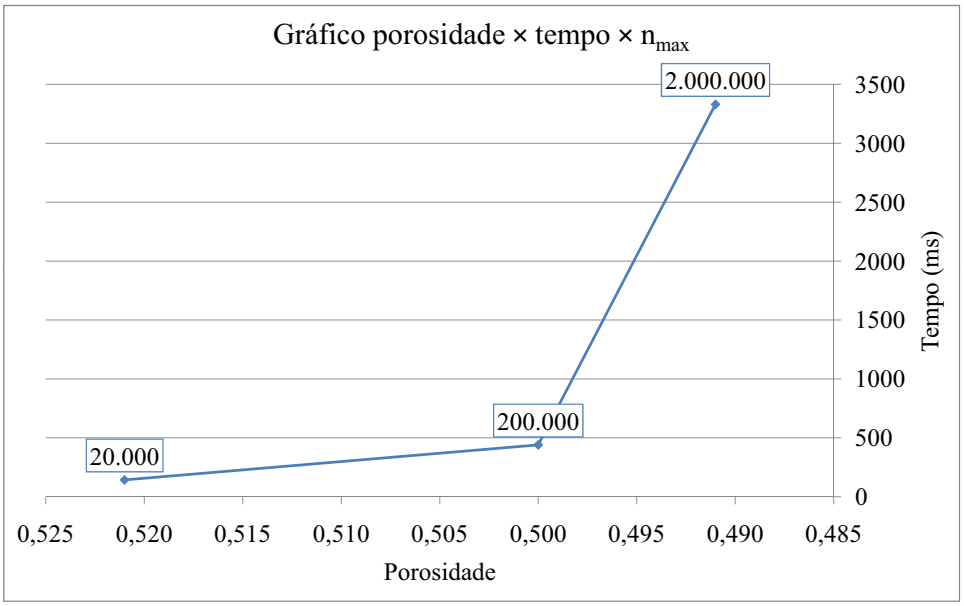

Figura 3.22: Comparação da geração por tentativas no programa PFC2D para diferentes n_{max}.

Geração por expansão do raio no programa PFC2D

No método anterior, não era possível gerar um arranjo com uma dada porosidade, e pior, a porosidade alcançada não é razoável para a maioria das simulações no MED. Este próximo método — geração por expansão de raio — atinge uma porosidade prescrita através da criação das partículas com raios menores e subseqüente expansão gradual desses raios, buscando o equilíbrio do arranjo.

Os parâmetros necessários são: o domínio, a porosidade, o número de partículas n e a razão r entre o raio da maior partícula e da menor. A razão foi naturalmente estabelecida — baseada na granulometria — como $r = 1,295/1,200$, porém o número de partículas traz um desafio — é preciso estimar quantas partículas são precisas para se chegar àquela dada porosidade, tendo-se em vista uma granulometria desejada. Uma opção é calcular um número de partículas aproximado baseado nos dados de entrada. Neste caso tem-se:

$$d_{med} = \frac{1,200 + 1,295}{2} \approx 1,250$$

$$A_{p,med} = \pi \cdot \frac{d_{med}^2}{4} = 1,23$$

$$A_d = w \cdot l = 8,6 \cdot 62,5 = 537,5$$

$$n_{aprox} = A_d / A_{p,med} \approx 438$$

O resultado pode ser visto na tabela 3.10.

Tabela 3.10: Resultado da geração por expansão do raio no programa PFC2D — 438 partículas.

Tempo(ms)	Diâmetro mínimo (mm)	Diâmetro máximo (mm)
520	1,058	1,142
580	1,057	1,142
700	1,058	1,142

Para fins de comparação, como um número de partícula médio pode ser inferido do resultado da geração através do algoritmo proposto — tabelas 3.5 e 3.6 —, este valor será utilizado em outro teste, sendo igual a 338.

Os resultados podem ser vistos na tabela 3.11 e um exemplo é mostrado na figura 3.23.

Tabela 3.11: Resultado da geração por expansão do raio no programa PFC2D — 338 partículas.

Tempo(ms)	Diâmetro mínimo (mm)	Diâmetro máximo (mm)
510	1,205	1,300
520	1,202	1,297
440	1,205	1,301

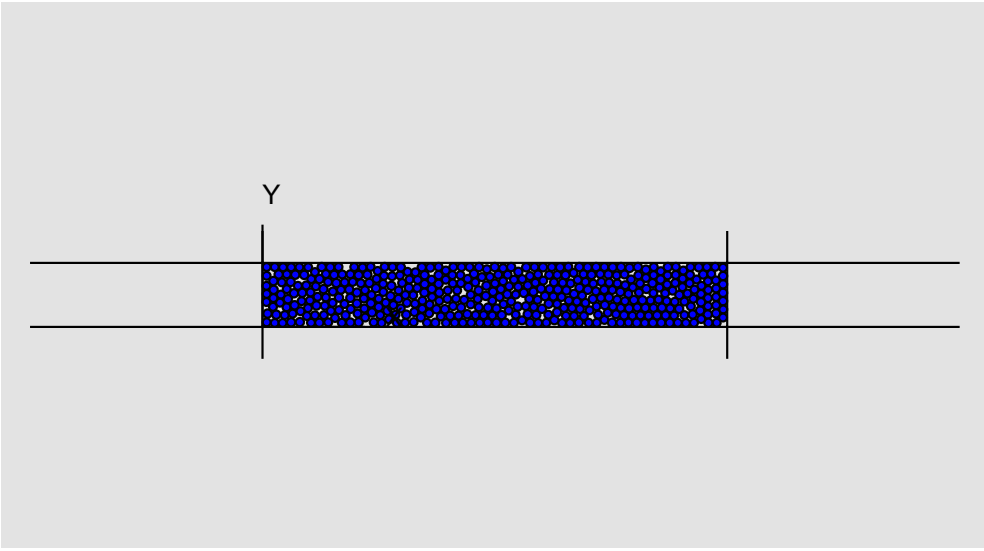

Figura 3.23: Resultado da geração por expansão do raio no programa PFC2D — 338 partículas.

Geração por repulsão explosiva no programa PFC2D

No método de geração por repulsão explosiva, as partículas são criadas aleatoriamente no domínio, já com seus raios finais, e em número suficiente para atingir a porosidade desejada. A sobreposição é permitida, e no caso de ser grande, assim também serão as forças. Tais forças podem gerar velocidades iniciais suficientemente altas, o que permite que algumas partículas escapem através das paredes do domínio. Para prevenir este tipo de acontecimento, durante os primeiros ciclos da convergência para o equilíbrio, várias vezes a energia cinética é reduzida a zero.

Os parâmetros necessários para este método são: o domínio, a granulometria, a porosidade e o número máximo de partículas. Este número máximo é necessário para prevenir, no caso de erro nos dados, que muitas partículas sejam geradas. Este número máximo foi dado igual a 1000.

Os resultados podem ser vistos na tabela 3.12 e um exemplo, na figura 3.24.

Tabela 3.12: Resultado da geração por repulsão explosiva no programa PFC2D.

Tempo(ms)	Partículas geradas	Porosidade
420	340	0,225
430	341	0,226
420	340	0,226

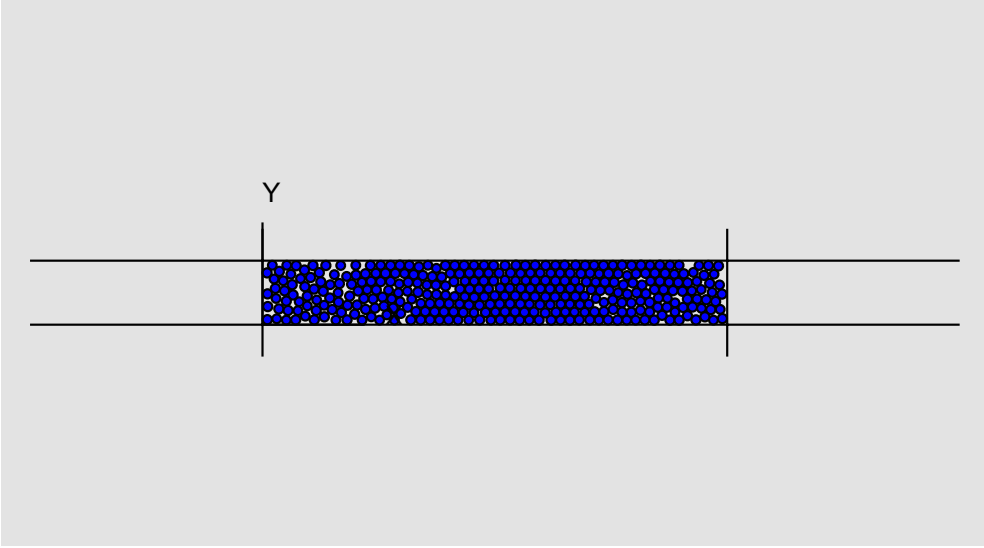

Figura 3.24: Resultado da geração por repulsão explosiva no programa PFC2D.

Considerações sobre a comparação

É importante que se faça algumas considerações sobre os três métodos de geração do PFC2D em comparação com o algoritmo de geração proposto:

- com o algoritmo de geração por tentativas, a porosidade obtida é muito elevada;
- com o algoritmo de geração por expansão do raio, não é possível, efetivamente, prescrever uma granulometria;
- com o algoritmo de geração por repulsão explosiva, embora do ponto de vista puramente de geração seja o melhor dos três métodos, pois gera um arranjo com uma dada porosidade e uma dada granulometria, esse arranjo pode ser extremamente não uniforme em relação à porosidade local, além de possuir grandes forças internas [1].

Para o caso estudado — que possui dimensões modestas—, ambos os programas — o desenvolvido neste projeto e o PFC2D — geram arranjos com rapidez, entretanto, o PFC2D necessita de uma maior interação com o usuário, incluindo a difícil conjectura de alguns parâmetros. Além disso, dependendo do método, alguns efeitos colaterais que devem ser levados em consideração podem surgir.

3.2.2 Investigação do desempenho

De modo a verificar genericamente o desempenho do algoritmo, foi testada uma série de configurações crescente de domínios quadrados. A granulometria segue a distribuição dada pela tabela 3.4 e é mostrada na figura 3.19. Os resultados são apresentados na tabela 3.13 e no gráfico 3.25.

Para testar a implementação do algoritmo em relação a um grande número de partículas, naturalmente é necessário aumentar a razão entre a área do domínio e a área média das partículas. Os testes aqui executados foram com grandes dimensões. Os resultados são mostrados na tabela 3.14.

Um teste em condições reais se faz necessário também, de maneira que não se perca o objetivo real de um algoritmo desse tipo. Para isso, dados de fraturas de poços reais da Petrobras foram obtidos do trabalho de Cachay [9]. A geometria do modelo de fratura adotada neste trabalho é apresentado na figura 3.26. Os dados de fraturas dos poços escolhidos para a análise e os resultados podem ser vistos nas tabelas 3.15 e 3.16, respectivamente.

Tabela 3.13: Resultado da geração de partículas em domínios quadrados.

Dimensões (mm × mm)	Tempo(ms)	Partículas geradas	Porosidade
10 × 10	109	54	0,340
	109	54	0,336
	110	53	0,346
12 × 12	110	80	0,322
	141	80	0,322
	125	81	0,313
15 × 15	156	135	0,264
	172	132	0,279
	156	135	0,264
20 × 20	234	243	0,252
	218	249	0,235
	204	245	0,245
50 × 50	407	1.624	0,202
	312	1.634	0,196
	313	1.616	0,207
100 × 100	640	6.610	0,187
	609	6.566	0,193
	641	6.675	0,180
120 × 120	797	9.542	0,186
	766	9.514	0,188
	812	9.558	0,184
150 × 150	1.141	14.810	0,191
	1.110	14.898	0,186
	1.110	14.905	0,186
200 × 200	1.656	26.546	0,184
	1.688	26.556	0,184
	1.734	26.593	0,183
500 × 500	9.344	166.642	0,181
	9.469	166.875	0,180
	9.578	169.416	0,167
1000 × 1000	37.563	674.027	0,171
	37.078	668.268	0,179
	37.235	667.820	0,179

3.3 Exemplos suplementares

São apresentados agora alguns exemplos que demonstram as possibilidades de aplicação da implementação do algoritmo proposto em domínios de formatos diversos.

As figuras 3.27 e 3.28 apresentam a geração de arranjos em um domínio trapezoidal de altura 20mm, e bases 10mm e 20mm. A primeira utiliza granulometria constante igual a 1,2mm e atinge uma porosidade de 0,233, enquanto a segunda utiliza granulometria uniformemente distribuída entre 1,2 e 2,4 e atinge uma porosidade de 0,283.

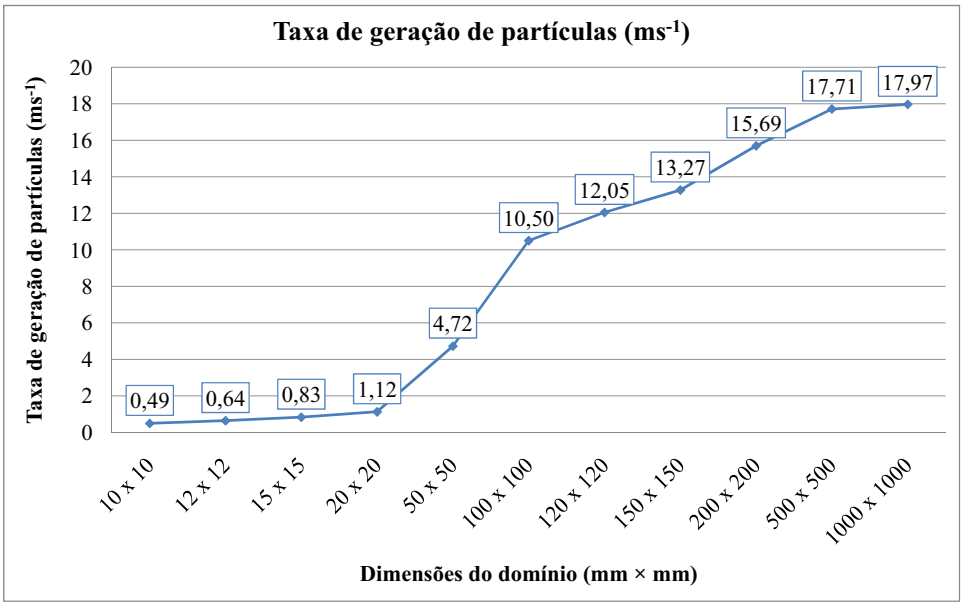

Figura 3.25: Taxa de geração de partículas.

Tabela 3.14: Resultado da geração de partículas em grandes dimensões.

Dimensões (mm × m)	Tempo(ms)	Partículas geradas	Porosidade
10 × 100	40.516	640.684	0,213
	40.860	640.668	0,213
	40.531	640.691	0,213
12 × 100	52.250	821.904	0,158
	52.078	783.259	0,198
	52.562	790.121	0,191
15 × 100	63.187	961.862	0,212
	62.484	961.868	0,212
	63.094	961.907	0,212

Tabela 3.15: Dados de fraturas dos poços da Petrobras.

Poço	Largura (mm)	Altura (m)
7–CP–0370–SE	2,6	8,0
7–CP–1414–SE	2,3	25,0
7–CP–0037–SE	4,2	17,0

As figuras 3.29 e 3.30 apresentam a geração de arranjos em um domínio trapezoidal invertido em relação ao anterior, mais ainda de altura 20mm, e bases 10mm e 20mm. A primeira utiliza granulometria constante igual a 1,2mm e atinge uma porosidade de 0,233, enquanto a segunda utiliza granulometria uniformemente distribuída entre 1,2 e 2,4 e atinge uma porosidade de 0,283.

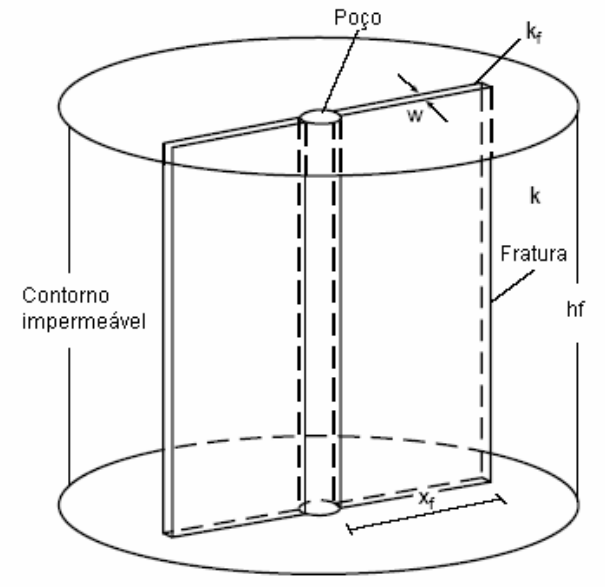

Figura 3.26: Geometria do modelo de fratura adotada (*apud* Economides e Nolte [17]).

Tabela 3.16: Resultado da geração de partículas em poços da Petrobras.

Poço	Tempo(ms)	Partículas geradas	Porosidade
7–CP–0370–SE	921	12.801	0,244
	906	12.795	0,244
	906	12.797	0,244
7–CP–1414–SE	1906	31.823	0,320
	1875	31.677	0,323
	1875	31.759	0,321
7–CP–0037–SE	2421	40.800	0,298
	2437	40.797	0,298
	2453	40.805	0,298

As figuras 3.31 e 3.32 apresentam a geração de arranjos em um domínio circular de raio 12,2mm. A primeira utiliza granulometria constante igual a 1,2mm e atinge uma porosidade de 0,277, enquanto a segunda utiliza granulometria uniformemente distribuída entre 1,2 e 2,4 e atinge uma porosidade de 0,300.

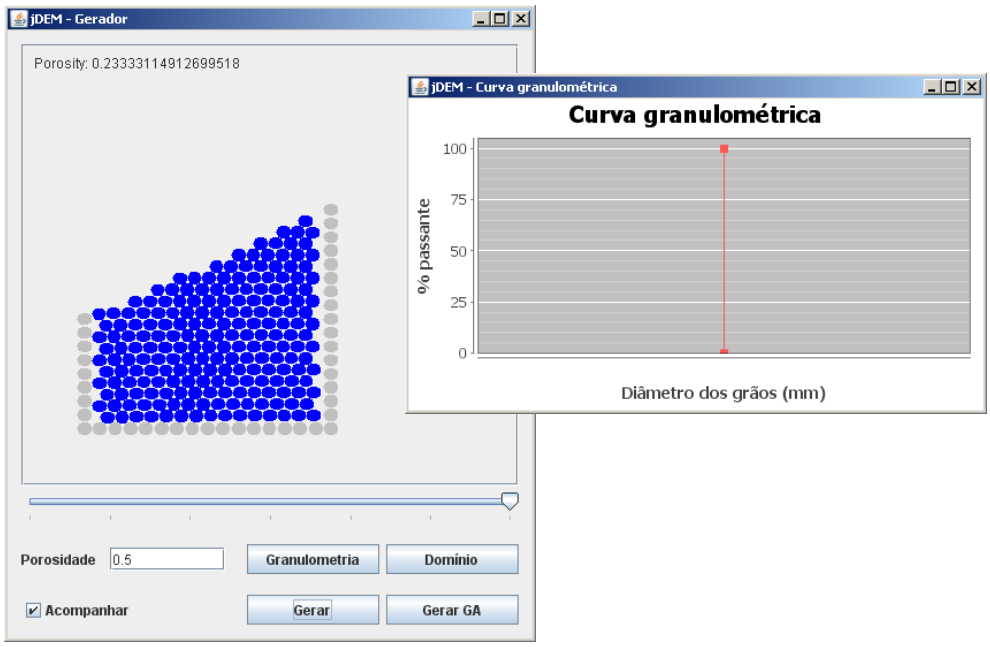

Figura 3.27: Geração de partículas em domínio trapezoidal com granulometria constante.

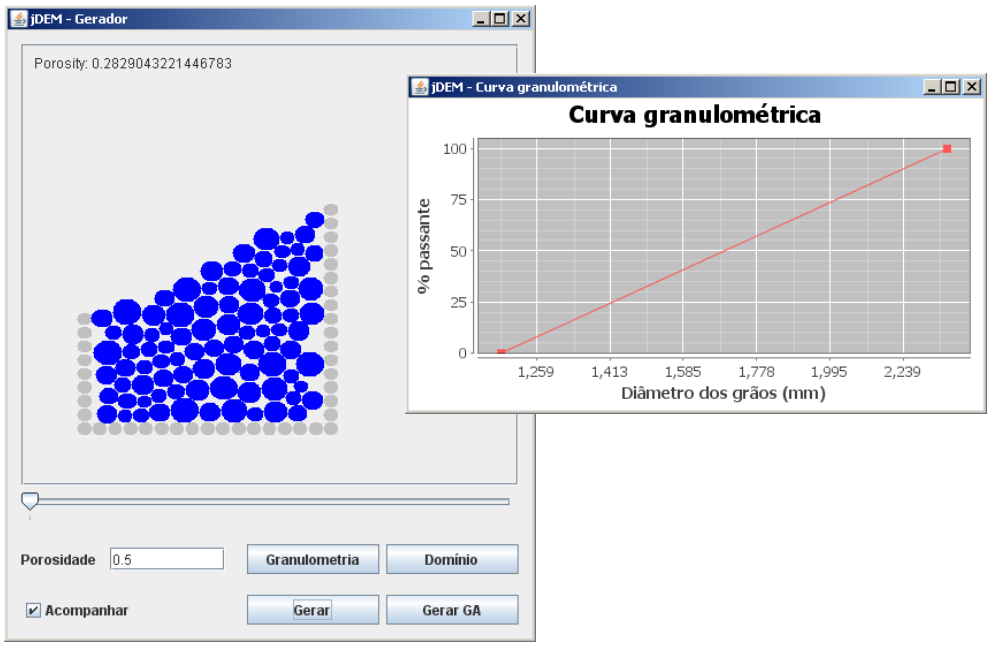

Figura 3.28: Geração de partículas em domínio trapezoidal com granulometria uniformemente distribuída.

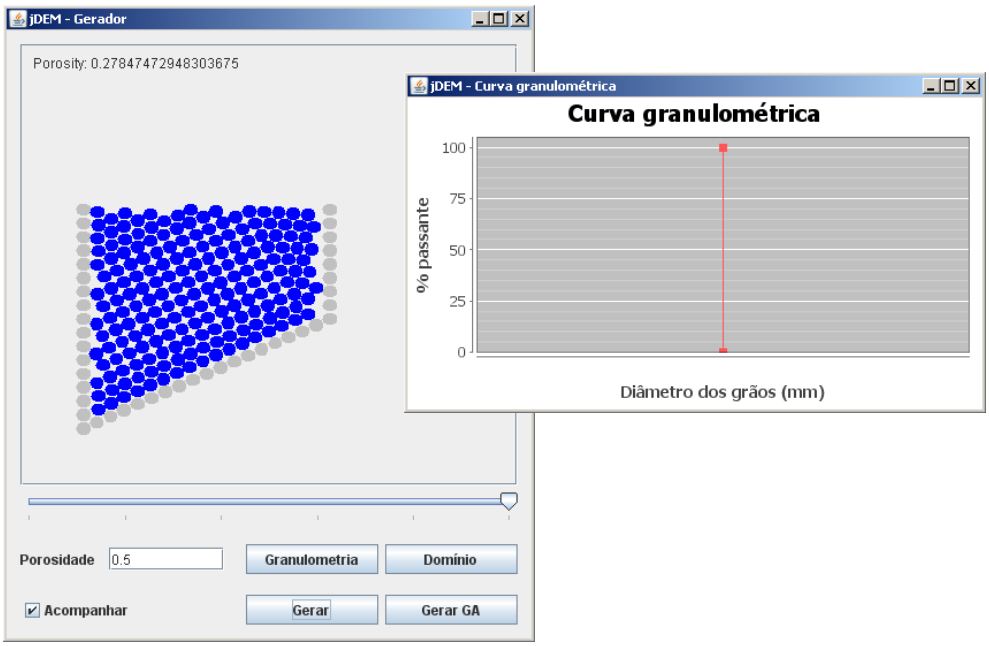

Figura 3.29: Geração de partículas em domínio trapezoidal invertido com granulometria constante.

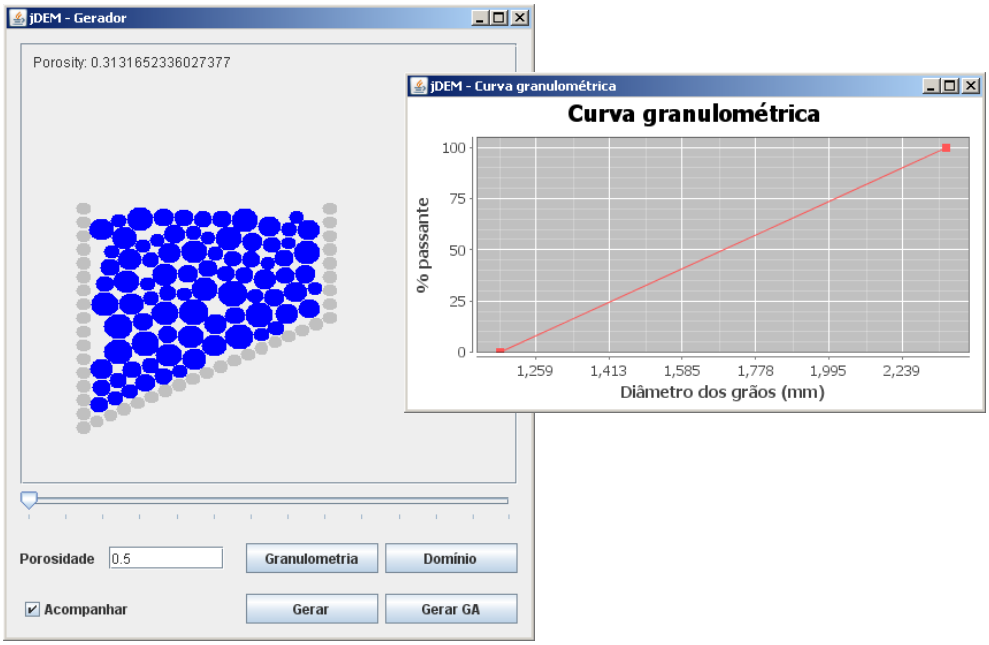

Figura 3.30: Geração de partículas em domínio trapezoidal invertido com granulometria uniformemente distribuída.

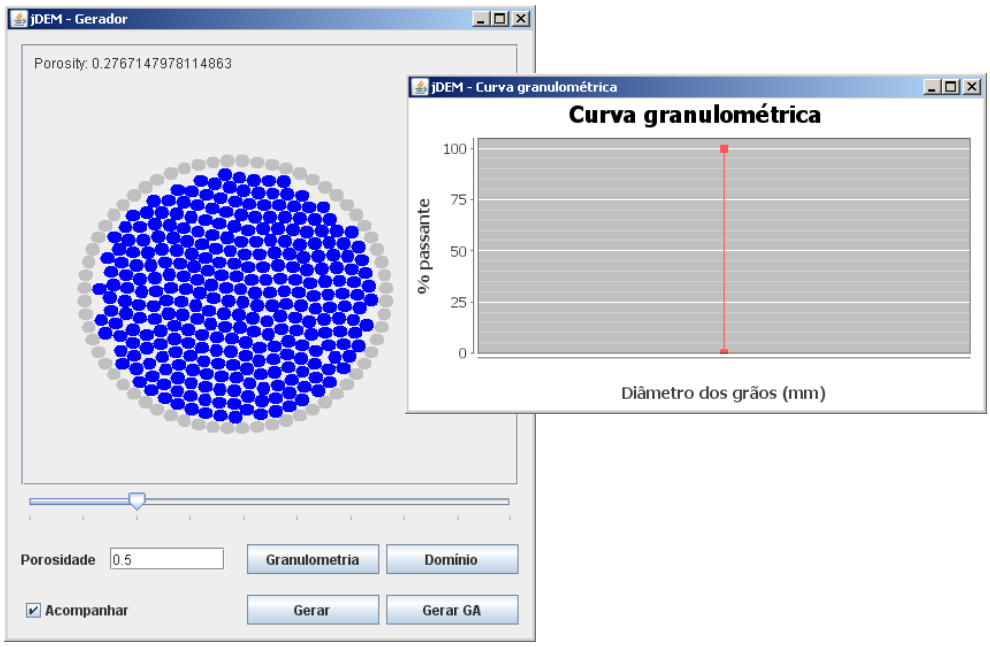

Figura 3.31: Geração de partículas em domínio circular com granulometria constante.

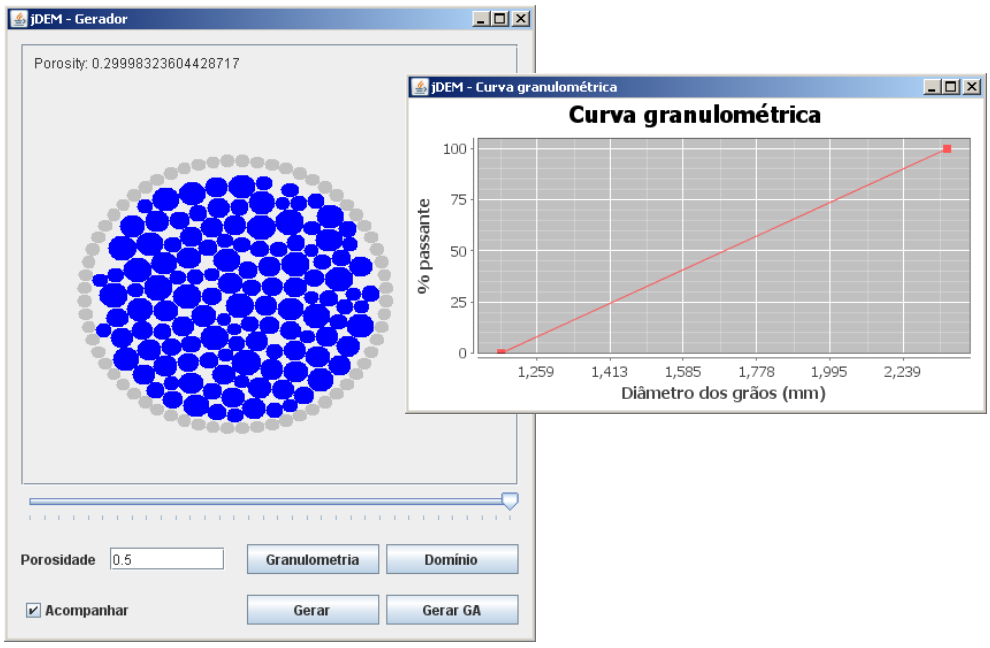

Figura 3.32: Geração de partículas em domínio circular com granulometria uniformemente distribuída.

4
Otimização de parâmetros dos arranjos gerados

Nem sempre o algoritmo escolhido para a realização de uma etapa do projeto retorna aquilo que exatamente seria necessário para a etapa seguinte. Em tais situações uma otimização é vantajosa, visando um melhor ajuste ao subseqüente desenrolar do trabalho. Deve-se levar em consideração ao otimizar um conjunto de parâmetros que, indubtavelmente, haverá algumas restrições que não poderão ser violadas. Neste projeto a otimização foi realizada utilizando-se algoritmos genéticos (AG).

4.1
Algoritmo genético

Chamamos algoritmos evolucionários (AE) um ramo na área da inteligência artificial (IA) que utiliza alguns mecanismos inspirados na teoria da evolução das espécies de Charles Darwin [13]. Na evolução segundo Darwin, características individuais são passadas de pais para filhos. Se um indivíduo é mais adaptado ao ambiente, é mais provável que sobreviva e que produza mais descendentes, passando suas características para a próxima geração. Tal fenômeno é chamado seleção natural.

Existem vários tipos de AE, todos compartilhando os mesmos fundamentos básicos, diferindo somente em alguns detalhes de implementação e natureza do problema. Dentre os AE, destacam-se os algoritmos genéticos (AG), como o proposto por Holland [20], que é aplicado neste projeto. De maneira geral, o AG pode ser definido como uma ferramenta computacional, pertencente ao grupo dos AE, que visa encontrar soluções aproximadas ou exatas a problemas de busca e otimização. O algoritmo é ilustrado na figura 4.1, que é melhor explicado nas seções seguintes.

4.1.1
Representação

O primeiro passo de um AG é a escolha da representação. Uma boa representação deve buscar expressar objetivamente a solução do problema em questão, além de ser manipulável computacionalmente.

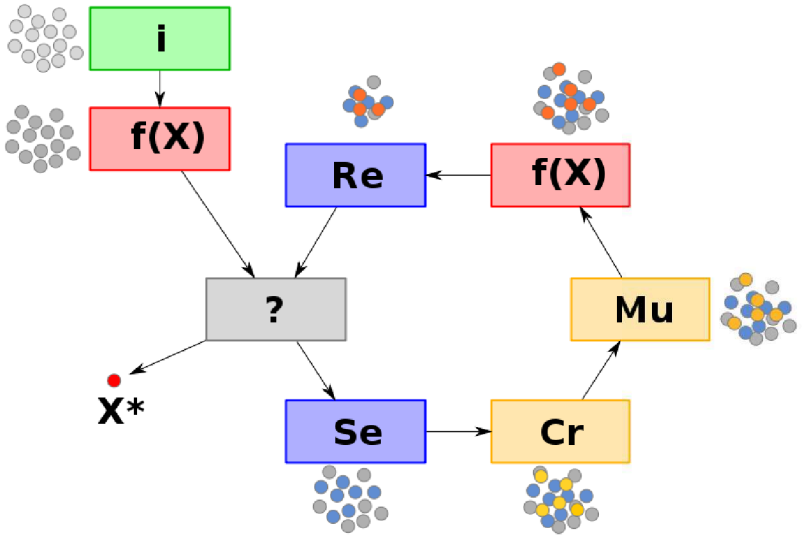

Figura 4.1: (Dreo [16])

Na representação de um indivíduo, cada característica importante do problema é chamada gene e o conjunto dos genes é chamado cromossomo. Cada cromossomo representa uma solução potencial do problema; sendo chamada população, o conjunto de cromossomos em uma determinada geração. A população inicial está sendo representada na figura 4.1 por i.

Na formulação original do AG, essa codificação somente é composta de valores binários, ou seja, cada gene é discretizado em bits dependendo da precisão que se deseja obter. Esses bits são chamados alelos e, naturalmente, não é necessário que cada gene tenha o mesmo número alelos.

Hoje em dia, entretanto, existem formulações que permitem que os genes possam representar diretamente parâmetros através de inteiros ou reais.

4.1.2 Avaliação

O conceito de seleção natural no contexto do AG depende de um critério de identificação dos indivíduos mais aptos em uma dada população. Esse critério, chamado função de avaliação ou função-objetivo, deve ser explicitamente definido e retorna um valor numérico que simboliza o quão perto está a solução representada pelo indivíduo da solução ideal. A função de avaliação é específica para cada problema que se deseja modelar. Está sendo representado na figura 4.1 por $f(X)$.

4.1.3 Operadores genéticos

Os operadores genéticos são responsáveis pela geração de novos indivíduos, sendo assim ferramentas necessárias para o processo de evolução. Os mais empregados serão melhor analisados adiante.

Seleção

Os operadores de seleção são responsáveis por escolher quais indivíduos terão a oportunidade de se reproduzir. Esta seleção deve levar em consideração a função de avaliação, de maneira que os indivíduos mais aptos tenham uma maior probabilidade de reprodução, sem, entretanto, desconsiderar completamente a participação dos indivíduos menos aptos. Os operadores de seleção podem ser identificados na figura 4.1 por *Se*.

Seleção por roleta Um dos operadores de seleção mais utilizados é a seleção por roleta, desenvolvida por De Jong [14]. Neste algoritmo, cada indivíduo na população recebe uma fatia de uma roleta, de tamanho proporcional a sua aptidão, como visto figura 4.2. Um valor é sorteado e o indivíduo correspondente é selecionado.

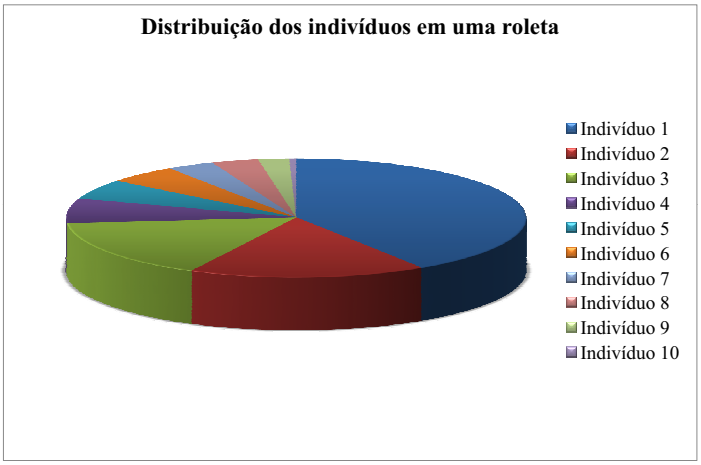

Figura 4.2: Exemplo de distribuição dos indivíduos em uma roleta.

Crossover

O *crossover*, também chamado recombinação ou cruzamento, opera nos indivíduos selecionados para reprodução, dois a dois. É, portanto, uma es-

tratégia de reprodução sexuada. Sua função é misturar o código genético dos indivíduos, na expectativa de gerar uma nova população com indivíduos mais aptos. Está sendo representado na figura 4.1 por Cr. Deve-se salientar que nem sempre os indivíduos selecionados sofrem recombinação — existe uma taxa de *crossover* que exprime a probabilidade deste ocorrer. Um cuidado que se deve ter é que ao ocorrer a recombinação, os descendentes não sejam indivíduos inválidos. No caso dos operadores de *crossover* possuirem alguma possibilidade de gerar tais indivíduos, será necessária uma validação, e em caso de invalidez, outros indivíduos deverão ser gerados. Dependendo da maneira que o problema é abordado, essa etapa pode ser muito custosa. Há uma infinidade de algoritmos de *crossover*, específicos para cada representação. Alguns dos mais comuns são descritos a seguir.

– Alguns tipos de *crossover* podem ser empregados independentemente da codificação ser binária ou não:

  De um ponto Utiliza um ponto de corte nos cromossomos. Este corte é aleatório e separa os indivíduos em duas partes. Essas partes são então permutadas entre os dois cromossomos, como na figura 4.3.

  De dois pontos Análogo ao de um ponto de corte, com a evidente diferença de ter dois pontos de corte como na figura 4.4. Os segmentos centrais são permutados.

– Outros tipos, no entanto, somente podem ser empregados no caso de codificações não-binárias:

  Aritmético Este *crossover* gera um descendente que tem seus genes definidos como sendo a média aritmética dos genes de seus genitores.

Mutação

A mutação é uma estratégia de reprodução assexuada que que provoca uma alteração aleatória em alguns genes, embora geralmente somente em um deles. Pode ser visto na figura 4.1 representado por Mu. Com freqüência, a mutação é aplicada a um ou ambos os descendentes gerados no *crossover*, segundo uma dada probabilidade. Deve-se ter o mesmo cuidado com a mutação que se tem com o *crossover* em relação à validade dos indivíduos. Existem inúmeros operadores de mutação. O mais básico simplesmente modifica aleatoriamente um gene do cromossomo, como na figura 4.5. Outro operador bem simples troca dois alelos de posição.

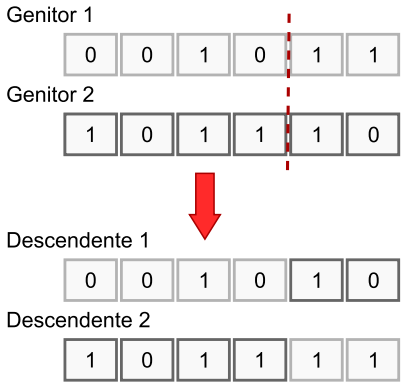

Figura 4.3: Exemplo de crossover de um ponto.

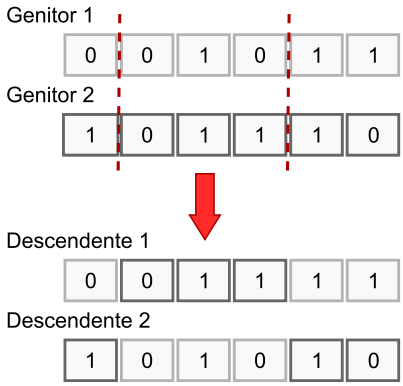

Figura 4.4: Exemplo de crossover de dois pontos.

Figura 4.5: Exemplo de mutação.

Reposição

Os algoritmos de reposição, representados na figura 4.1 por Re, não existiam na formulação original de Holland [20]. Nessa formulação, os operadores genéticos de *crossover* e mutação atuam em uma população temporária, e os descendentes dessas operações são adicionados à população, que após ser completamente preenchida se torna a população atual. Outras formulações permitem que esses operadores sejam aplicados diretamente à população genética. Nesse caso, de modo a manter o tamanho da população constante, não se pode simplesmente acrescentar-lhe os descendentes — primeiramente deve-se retirar um igual número de indivíduos da população. Podem ser vistos alguns tipos abaixo.

Reposição dos genitores Os genitores dão espaço aos descendentes.

Reposição dos piores indivíduos Aleatoriamente selecionam-se dentre os piores indivíduos os indivíduos a serem eliminados. Geralmente são considerados os últimos 10% [38].

Reposição aleatória São selecionados aleatoriamente os indivíduos que serão excluídos.

4.1.4 Critérios de parada

Idealmente, o AG deveria parar quando alcançasse a solução ótima para o problema estudado (X^* na figura 4.1), no entanto, nem sempre essa solução é conhecida, e portanto, outros critérios de parada devem ser aplicados. Os critérios de parada são representados na figura 4.1 por ? (sinal de interrogação). Alguns dos critérios normalmente usados são:

- alcançar um nível satisfatório de aptidão;
- alcançar um número máximo de gerações;
- a convergência da população. Por exemplo, se 95% da população possui os mesmos genes [38];
- atingir o orçamento atribuído (tempo de computação / dinheiro);
- combinação dos critérios anteriores.

Dependendo do critério de parada, ao final do algoritmo, pode acontecer de não ter sido alcançada uma solução satisfatória.

4.1.5
Implementação do AG no projeto

Para a implementação do AG no projeto, foi utilizada uma biblioteca chamada Java Genetic Algorithms Package (JGAP). Segundo o autor [31], essa biblioteca "provê mecanismos genéticos básicos que podem ser facilmente usados para aplicar princípios evolucionários a soluções de problemas, sendo projetada para ser muito fácil para o uso 'out of the box' (imediato), ainda que altamente modular, permitindo que usuários mais 'aventureiros' possam facilmente implementar operadores genéticos e outros subcomponentes personalizados".

A implementação do AG é direta utilizando o JGAP. Além de configurações básicas, é necessário apenas definir a função de avaliação, algo que é inerente a cada problema.

4.2
Otimização do arranjo de partículas

No caso da geração de arranjos para a simulação através do MED, muitas vezes é interessante modelar o mais realisticamente possível um dado problema. Em casos assim, pode ser necessário prescrever um valor de porosidade, por exemplo. No entanto, o algoritmo proposto não permite esta parametrização e por isso, foi utilizado o AG para otimizar a porosidade.

A porosidade é definida pela razão entre volume de vazios e volume total, como mostrado na equação 4-1.

$$\phi = \frac{V_V}{V_T} \tag{4-1}$$

No projeto, por se tratar de uma representação bidimensional, a porosidade foi definida como a razão entre a área total subtraída da área somada de todas as partículas e a área total, como mostrado na equação 4-2.

$$\phi = \frac{A_T - \sum A_p}{A_T} \tag{4-2}$$

4.2.1
Configuração do AG

Para o caso de otimização da porosidade, pode-se ou modificar o tamanho das partículas do arranjo, sem que, no entanto, se desvie da granulometria prescrita, ou reordenar as partículas do arranjo.

A formulação inicialmente imaginada para representar o arranjo foi um cromossomo composto por uma quantidade n_{est} de genes. Cada gene representaria o tamanho de uma partícula; e n_{est} seria um número suficientemente

grande para que qualquer indivíduo fosse capaz de preencher o domínio, conforme a equação 3-1.

Entretanto, com esta representação, seria muito difícil obedecer à granulometria dada, sobrecarregando a etapa de validação dos indivíduos.

Uma maneira possível para contornar este problema é a utilização do método da transformada inversa como visto em Devroye [15]. Este método permite a geração de números aleatórios não-uniformes através da função de distribuição acumulada: dada uma variável contínua uniforme $U \in [0, 1]$ e a função de distribuição acumulada F, pode-se construir uma variável aleatória $X = F^{-1}(U)$ que possui distribuição F.

A curva granulométrica é uma função de distribuição acumulada: no eixo das abcissas são indicados os diâmetros dos grãos e no das ordenadas a porcentagem passante acumulada em cada peneira, embora fosse igualmente possível utilizar a porcentagem retida acumulada. Como a curva adotada é composta por segmentos de retas, a inversão da função de distribuição acumulada em cada trecho é trivial, simplesmente sendo realizada uma regra de três, observando-se que o resultado deve ser tratado devido ao gráfico ser semilogarítmico.

Agora, os genes não mais representam os tamanhos das partículas, e sim, o percentual passante, que é convertido em tamanho durante o processo.

A função de avaliação foi modelada utilizando o algoritmo proposto neste projeto para geração de arranjos. Durante a avaliação da aptidão de cada cromossomo, é gerado um arranjo através do algoritmo proposto, e desse arranjo calcula-se a porosidade. O problema será considerado como de minimização, ou seja, um cromossomo é considerado mais apto quanto menor for o resultado da função de avaliação. Neste problema, o resultado de tal função é o quadrado da diferença entre a porosidade avaliada e a porosidade desejada.

4.3 Otimização de propriedades das partículas

Além da otimização do arranjo de partículas em si, outra sugestão que pode ser feita é a otimização de propriedades das partículas visando gerar, por exemplo, arranjos para a simulação de materiais compósitos, como concreto (figura 4.6), e de materiais com gradação funcional (MGF).

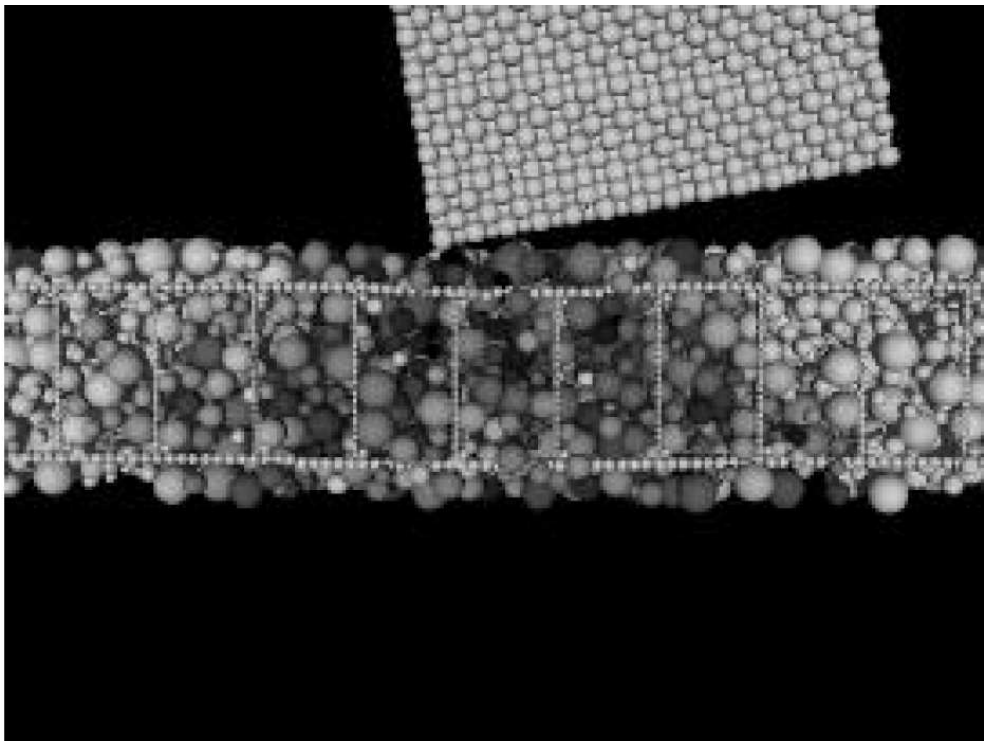

Figura 4.6: Modelagem de concreto armado no MED.

4.3.1 Material com gradação funcional

Freqüentemente em projetos de estruturas, aos materiais são atribuídas funções e/ou propriedades uniformizadas. Muitas aplicações, entretanto, estão expostas a condições adversas para as quais tais materiais não são apropriados, o que motiva o desenvolvimento de novos materiais com propriedades otimizadas. Inicialmente, foram desenvolvidos materiais compostos por camadas de diferentes características, todavia, o problema da descontinuidade entre as camadas é um fator limitante na utilização plena dessas características. De forma a suprir tais deficiências, foram desenvolvidas técnicas que visavam gerar materiais que tinham suas características modificadas gradualmente. Segundo Japan Aerospace Exploration Agency (JAXA) [23], define-se materiais com gradação funcional (MGF) como um material cujas composições e/ou funções variam continuamente ou segundo uma função-degrau, desde um lado até o outro.

Os MGF podem ser obtidos através de diferentes tipos de gradação microestrutural, como: a contínua, a discreta e a multi-fásica; conforme mostrado na figura 4.7.

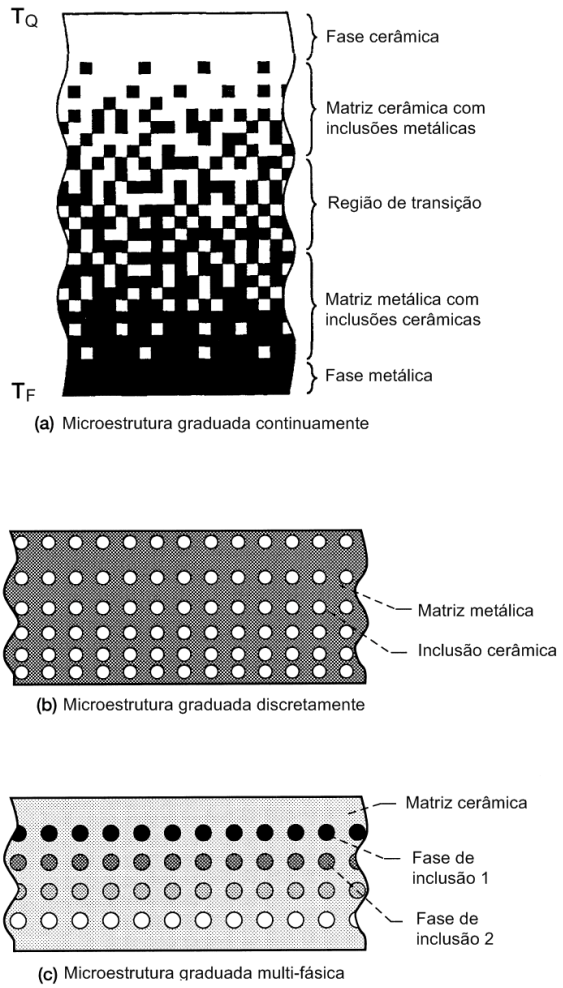

Figura 4.7: Exemplos de diferentes tipos de microestruturas de gradação funcional (Aboudi et al. [2]).

Há muitos campos de utilização dos MGF. A indústria aeronáutica — o campo de aplicação para o qual o MGF foi originalmente proposto — utiliza barreiras de tratamento térmico (figura 4.8). Esse MGF tem por finalidade proteger os componentes metálicos das altas temperaturas nas turbinas. No ramo dos materiais industriais buscam-se materiais que tenham tanto resistência ao desgaste quanto tenacidade. A optoeletrônica busca alcançar altas velocidades de transmissão através de MGF que possuam índice de refração variável. Em relação a biomateriais, uma área de estudo é a substituição de ossos e juntas. Nesse caso, os MGF necessitam de dureza e resistência à corrosão, além de compatibilidade biológica e inocuidade. Um exemplo é visto na figura 4.9.

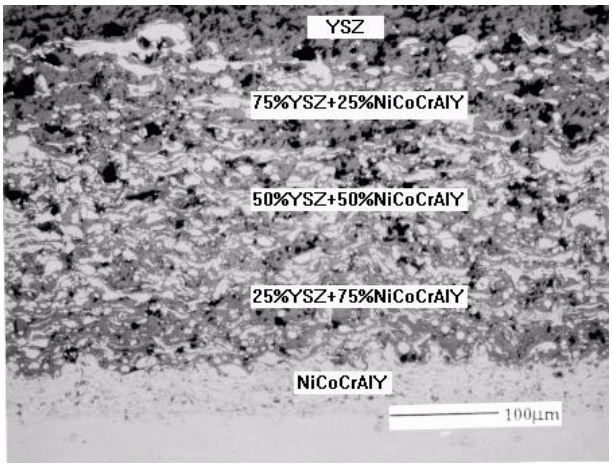

Figura 4.8: Barreira de tratamento térmico (AMPRSP [4]).

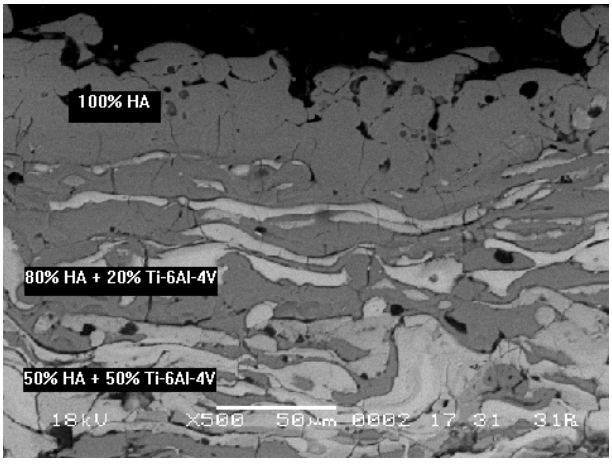

Figura 4.9: MGF para utilização em implantes (AMPRSP [5]).

4.3.2 Configuração do AG

Durante a otimização das propriedades das partículas, a configuração física do arranjo não é alterada, sendo o arranjo inicialmente gerado com a utilização do algoritmo proposto.

O cromossomo é composto de genes que representam as propriedades a serem otimizadas.

A função de avaliação foi modelada utilizando o algoritmo proposto neste projeto para geração de arranjos. Durante a avaliação da aptidão de cada cromossomo, é gerado um arranjo através do algoritmo proposto, e desse arranjo calcula-se a porosidade. O problema será considerado como de minimização, ou seja, um cromossomo é considerado mais apto quanto menor for o resultado da função de avaliação. Neste problema, o resultado de tal função é o quadrado da diferença entre a porosidade avaliada e a porosidade desejada.

4.4 Avaliação da otimização

De modo a avaliar a otimização através do AG, foram realizados alguns testes. Para efeito de simplificação, foi definido um domínio quadrado de 20mm de lado.

4.4.1 Avaliação da otimização para aumento da porosidade

A fim de simular uma porosidade alta, utilizou-se um valor médio obtido pelo programa PFC2D para o método de tentativas. Essa porosidade foi estipulada como 0,5. Os outros parâmetros foram modificados e combinados de forma a conceber os seguintes testes.

O teste AP–1 (figura 4.10) tem granulometria constante igual a 1,2mm e portanto, não possui variação na configuração do arranjo em relação ao algoritmo rodado isoladamente. Este teste foi concebido principalmente para comparação dos tempos.

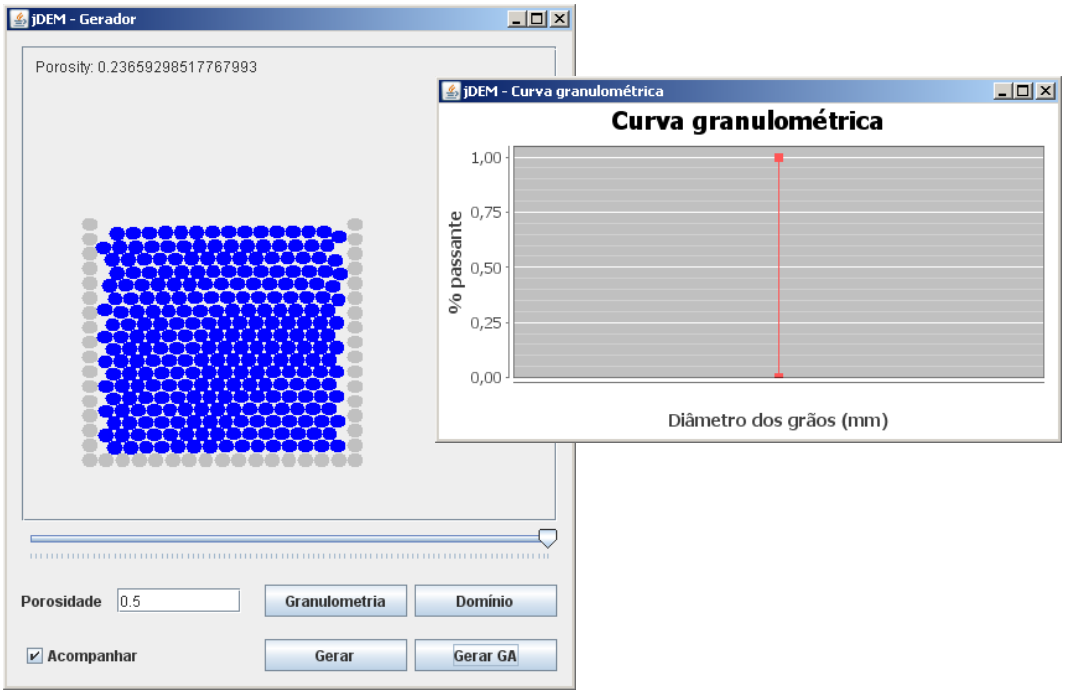

Figura 4.10: Teste AP–1 — granulometria constante igual a 1,2mm.

O teste AP–2 (figura 4.11) tem granulometria uniformemente distribuída entre 1,2mm e 1,3mm. Como quase não há variação na granulometria, obtém-se somente um leve aumento da porosidade.

O teste AP–3 (figura 4.12) tem granulometria uniformemente distribuída entre 1,2mm e 2,4mm. A porosidade já apresenta um aumento mais significativo nesse teste.

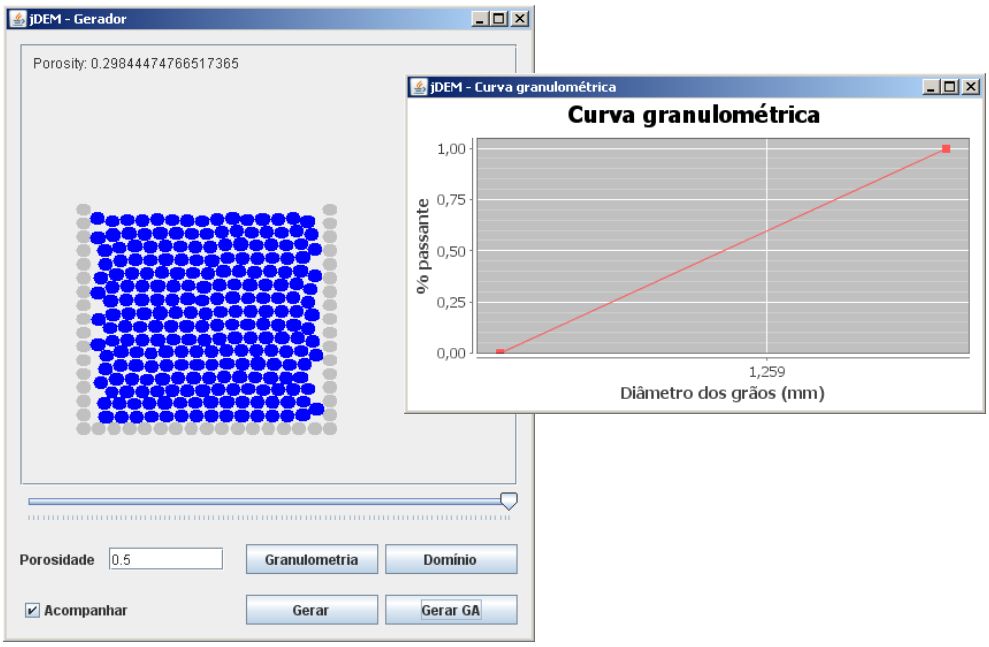

Figura 4.11: Teste AP–2 — granulometria uniformemente distribuída entre 1,2mm e 1,3mm.

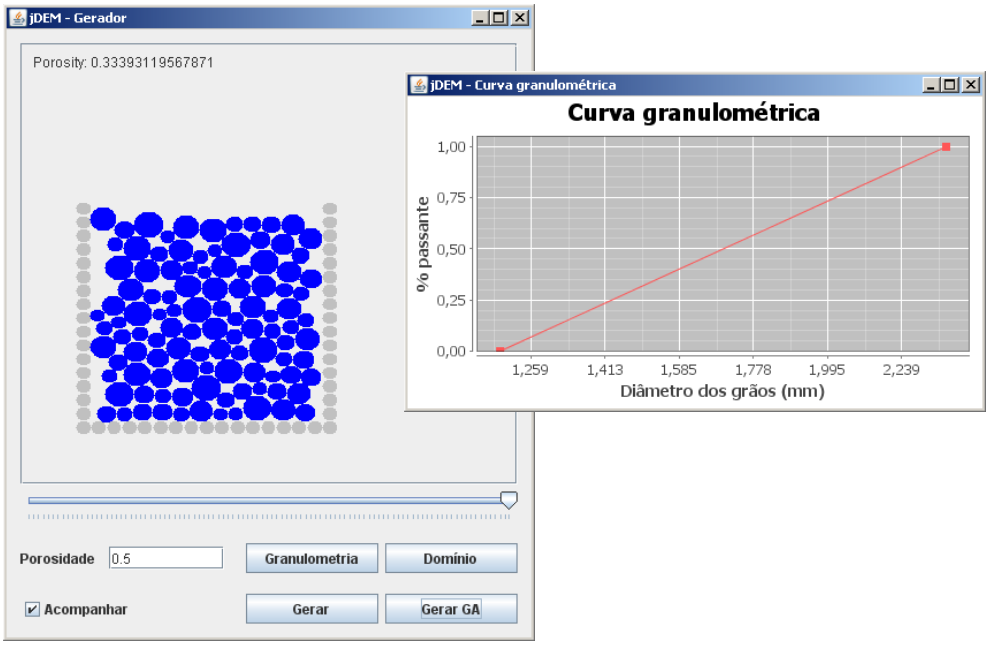

Figura 4.12: Teste AP–3 — granulometria uniformemente distribuída entre 1,2mm e 2,4mm.

O teste AP–4 (figura 4.13) tem granulometria uniformemente distribuída entre 1,2mm e 4,8mm. Embora exista uma maior variação da granulometria, ainda não foi possível alcançar uma porosidade alta como a desejada.

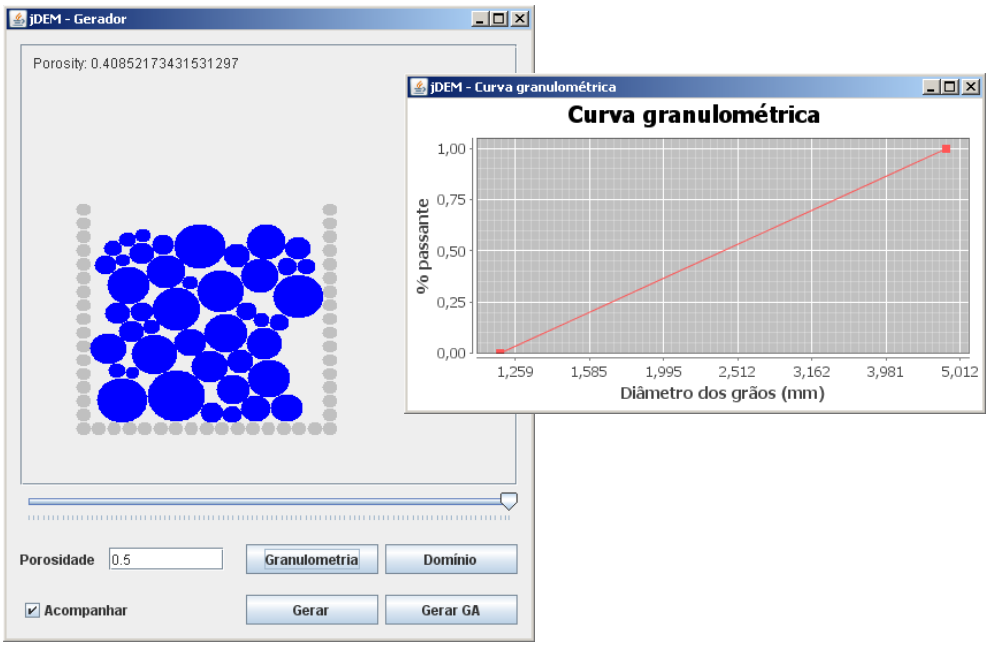

Figura 4.13: Teste AP–4 — granulometria uniformemente distribuída entre 1,2mm e 4,8mm.

O teste AP–5 (figura 4.14) tem granulometria estabelecida segundo a tabela 4.1. É preciso ser ressaltado que o conceito de porosidade já não é mais tão correto, dado que o pacote granular não ocupa completamente o domínio. Para que esta denominação continue a ser válida, deveriam ser desconsiderados os vazios que estão ao redor do pacote. Esse problema é minimizado conforme há um aumento das dimensões do domínio.

Tabela 4.1: Granulometria utilizada no teste AP–5.

% passante acumulada	Granulometria
0–10	uniformemente distribuída entre 1,2 e 2,4
10–90	constante igual a 2,4
90–100	uniformemente distribuída entre 2,4 e 6,0

A tabela 4.2 resume os resultados dos testes AP–1 a AP–5.

Tabela 4.2: Resultados da otimização em um domínio de dimensões 20mm × 20mm para os testes AP–1 a AP–5.

Teste	Porosidade alcançada	Tempo (s)
AP–1	0.237	609
AP–2	0.298	507
AP–3	0.334	352
AP–4	0.409	312
AP–5	0.467	240

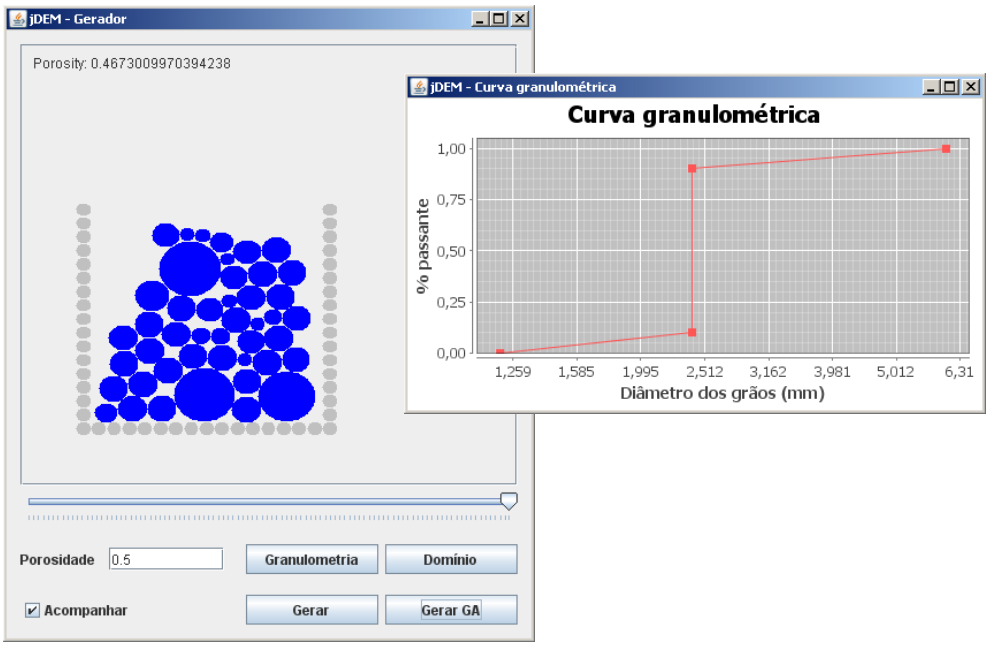

Figura 4.14: Teste AP–5 — granulometria estabelecida segundo a tabela 4.1.

4.4.2 Avaliação da otimização de propriedades das partículas

Para testar a otimização de propriedades das partículas foram feitos os seguintes testes variando a cor das partículas. Nota-se que poderia ser mudada esta propriedade por uma outra qualquer, somente sendo escolhida a propriedade cor para uma fácil identificação visual do resultado.

No teste MGF–1 (figura 4.15), o arranjo é constituído somente por partículas com uma mesma propriedade.

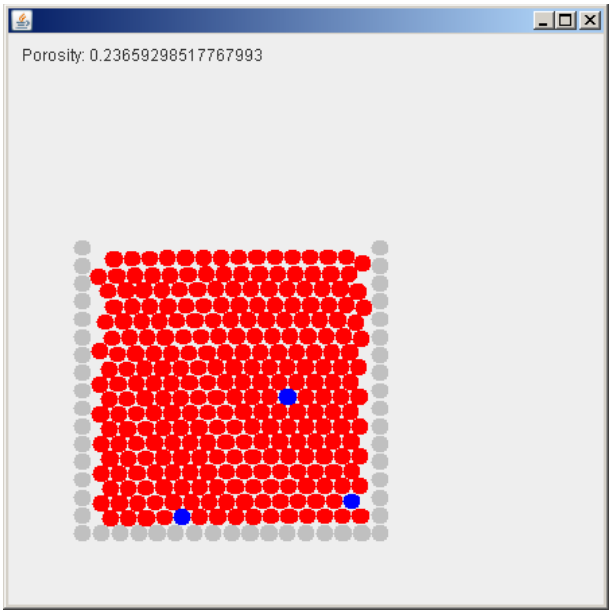

Figura 4.15: Teste MGF–1 — arranjo com uma propriedade.

No teste MGF–2 (figura 4.16), o arranjo é constituído por partículas com duas propriedades distribuídas em camadas sobrepostas.

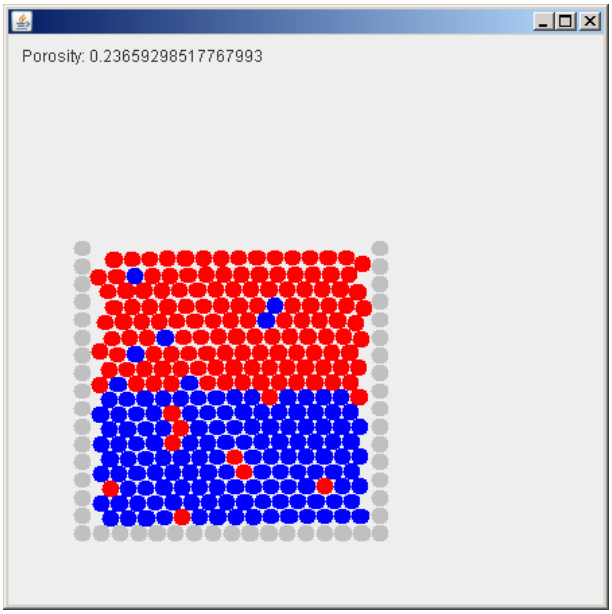

Figura 4.16: Teste MGF–2 — arranjo com duas propriedades distribuídas em camadas sobrepostas.

No teste MGF–3 (figura 4.17), o arranjo é constituído por partículas com três propriedades distribuídas em camadas sobrepostas.

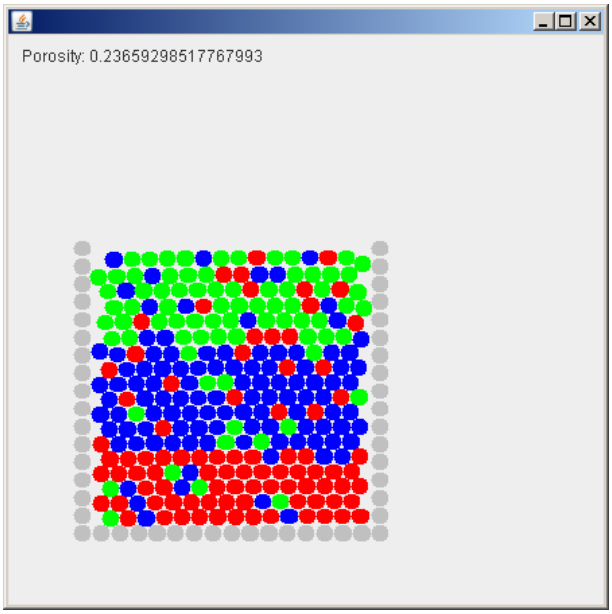

Figura 4.17: Teste MGF–3 — arranjo com três propriedades distribuídas em camadas sobrepostas.

A tabela 4.3 resume os resultados dos testes MGF–1 a MGF–3.

Tabela 4.3: Resultados da otimização em um domínio de dimensões 20mm × 20mm para os testes MGF–1 a MGF–3.

Teste	Tempo (s)
MGF–1	257
MGF–2	260
MGF–3	257

5
Conclusão

O objetivo principal deste trabalho foi o desenvolvimento de um algoritmo para geração de arranjos densos de partículas para utilização no método dos elementos discretos (MED). Nesse sentido, a implementação do algoritmo obteve resultados satisfatórios, tanto em relação ao tempo computacional quanto ao resultado final dos arranjos.

O algoritmo proposto comparado ao programa PFC2D reduziu o tempo computacional necessário para a geração dos arranjos em dois dos três métodos. Além disso, no único em que foi mais lento, a porosidade obtida foi até 47% menor.

O algoritmo funciona *out of the box*, ou seja, não é necessário ser definido nenhum outro parâmetro, além daqueles que descrevem o problema: granulometria e domínio.

O tempo de processamento tem uma melhora expressiva quando do emprego da *quadtree*, sendo reduzido em cerca de 67%.

A otimização da porosidade não é tão eficaz quanto o esperado inicialmente, pois só há alteração significativa quando existe grande variabilidade na granulometria, o que não acontece por exemplo com os materiais selecionados para sustentação de fraturas, os quais tem rigorosa seleção.

Por gerar arranjos de granulometria variada empregando um processo aleatório para determinação dos tamanhos das partículas, o algoritmo é indicado para modelar materiais como solo e agentes de sustentação de fratura.

Aplicações que necessitam de um arranjo pouco denso não são apropriadas para o algoritmo proposto, inclusive, não é possível gerar arranjos nos quais as partículas não estejam em contato umas com as outras.

5.1
Propostas para trabalhos futuros

Algumas propostas para trabalhos futuros são apresentadas a seguir:

– a implementação em três dimensões é um desdobramento natural do trabalho e, a princípio, seria de relativa simplicidade, para o caso de partículas esféricas;

- de modo a trabalhar com domínios cada vez maiores em sistemas com quantidade de memória limitante, considera-se a utilização de particionamento de domínio. É interessante implementar um gerenciador de particionamento dado que essa possibilidade já está prevista e a infraestrutura, preparada. Estes subdomínios podem ser utilizados para auxiliar a geração das partículas, ademais da avaliação local da configuração dos arranjos;
- o controle da granulometria pode ser melhorado, desenvolvendo-se uma verificação mais robusta para cada cromossomo;
- convém investigar com mais detalhes as vantagens que podem ser obtidas através de modificações na função de avaliação, como por exemplo, cromossomos que tenham mais genes aproveitados sejam considerados mais aptos;
- o emprego de diferentes estruturas de dados deve ser examinado em busca de reduções ainda maiores no tempo computacional.

Referências Bibliográficas

[1] **PFC2D User's Manual (Version 4.0)**. Minnesota. 2.1, 3.2.1, 3.2.1, 3.2.1

[2] ABOUDI, J.; M.-J., P. ; M., A. S. **High-order theory for functionally graded materials**. In: COMPOSITES PART B: ENGINEERING, volumen 30. Elsevier, 1999. 4.7

[3] ALVARADO, L. A. S. **Simulação bidimensional de corridas de detritos usando o método de elementos discretos**. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, June 2006. 1

[4] AMPSRP. **Functionally graded materials**. `http://www3.ntu.edu.sg/mae/Research/Programmes/Adv_Materials/FGM.htm`, July 2009. 4.8

[5] AMPSRP. **Thermal spraying coatings**. `http://www3.ntu.edu.sg/mae/Research/Programmes/Adv_Materials/Coatings.htm`, July 2009. 4.9

[6] ASGIAN, M.; CUNDALL, P. ; BRADY, B. **Mechanical stability of propped hydraulic fractures: A numerical study**. Journal of Petroleum Technology, 47:203–208, Mar. 1995. 3.2.1

[7] BAGI, K. **An algorithm to generate random dense arrangements for discrete element simulations of granular assemblies**. Granular Matter, 7(1):31–43, Apr. 2005. 2.3, 2.3

[8] BRATBERG, I.; RADJAI, F. ; HANSEN, A. **Dynamic rearrangements and packing regimes in randomly deposited two-dimensional granular beds**. Physical Review E, 66(3):031303, Sept. 2002. 2.1

[9] CACHAY, L. R. S. **Fluxo de partículas de sustentação em poços de petróleo estimulados por fraturamento hidráulico**. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, Dec. 2004. 3.2.2

[10] CARVALHO, JR., H.; CINTRA, D. T.; RAMOS, JR., A. S.; LAGES, E. N. ; RAMOS, V. C. L. **Utilização do método dos elementos discretos para a análise de dutos enterrados**. In: ANAIS DAS XXXII JORNADAS SULAMERICANAS DE ENGENHARIA ESTRUTURAL, p. 3467–3475, May 2006. 1

[11] CUI, L.; O'SULLIVAN, C. **Analysis of a triangulation based approach for specimen generation for discrete element simulations**. Granular Matter, 5(3):135–145, Dec. 2003. 2.3

[12] CUNDALL, P. A.; STRACK, O. D. L. **A discrete numerical model for granular assemblies**. Geotechnique, 29(1):47–65, 1979. 1

[13] DARWIN, C.; CARROLL, J. **On the Origin of Species**. Broadview Press, 2003. 4.1

[14] DE JONG, K. A. **An analysis of the behavior of a class of genetic adaptive systems**. Tese de Doutorado, Ann Arbor, MI, USA, 1975. 4.1.3

[15] DEVROYE, L. **Non-uniform random variate generation**. Springer-Verlag, New York, 1986. 4.2.1

[16] DRÉO, J. N. **Archivo:evolutionary algorithm.svg**. `http://es.wikipedia.org/wiki/Archivo:Evolutionary_algorithm.svg`, Feb. 2007. 4.1

[17] Economides, M. J.; Nolte, K. G., editors. **Reservoir Stimulation**. John Wiley & Sons, 3 edition, jun 2000. 3.26

[18] FENG, Y. T.; HAN, K. ; OWEN, D. R. J. **Filling domains with disks: an advancing front approach**. In: INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, volumen 56, p. 699–713, Department of Civil Engineering, University of Wales Swansea, Singleton Park, Swansea SA2 8PP, U.K., 2003. 2.1, 2.3

[19] FINKEL, R. A.; BENTLEY, J. L. **Quad trees a data structure for retrieval on composite keys**. Acta Informatica, 4(1):1–9, Mar. 1974. 3.1.1

[20] HOLLAND, J. H. **Adaptation in natural and artificial systems**. University of Michigan, Ann Arbor, MI, USA, 1975. 4.1, 4.1.3

[21] HUAMÁN, C. J. A. **Simulação 3d pelo método dos elementos discretos de refluxo de material de sustentação de fraturas em**

poços de petróleo. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, Oct. 2008. 3.2.1

[22] HUSTRULID, A. I. **Parallel implementation of the discrete element method**. Report, Engineering Division, Colorado School of Mines, Golden, CO, 80401, 1997. 1

[23] JAXA — JAPAN AEROSPACE EXPLORATION AGENCY. **FGMs-Database**. `http://fgmdb.kakuda.jaxa.jp/e_index.html`, jul 2009. 4.3.1

[24] JERIER, J.-F.; DONZE, F. ; IMBAULT, D. **An algorithm to generate random dense arrangements discs based on the triangulation**. In: DISCRETE ELEMENT GROUP FOR HAZARD MITIGATION, ANNUAL REPORT, volumen 3, p. D1–D7, 2007. 2.3

[25] JODREY, W. S.; TORY, E. M. **Computer simulation of close random packing of equal spheres**. Physical Review A, 32(4):2347–2351, Oct. 1985. 2.2

[26] JULLIEN, R.; MEAKIN, P. **Computer simulations of steepest descent ballistic deposition**. Colloids and Surfaces A: Physicochemical and Engineering Aspects, 165(1-3):405–422, 2000. 2.1

[27] KANSAL, A. R.; TORQUATO, S. ; STILLINGER, F. H. **Diversity of order and densities in jammed hard-particle packings**. Physical Review E, 66(4):041109, Oct. 2002. 2.1

[28] LUBACHEVSKY, B. D.; STILLINGER, F. H. **Geometric properties of random disk packings**. Journal of Statistical Physics, 60(5–6):561–583, Sept. 1990. 2.1

[29] MARKUS, A. T.; MIGUEL, L. F. F. **Simulação de fratura do úmero humano utilizando o método dos elementos discretos**. In: Alberto Cardona, Mario Storti, C. Z., editor, MECáNICA COMPUTACIONAL, volumen XXVII, p. 3411–3422, San Luis, Argentina, Nov. 2008. Asociación Argentina de Mecánica Computacional. 1

[30] MATUTTIS, H. G.; LUDING, S. ; HERRMANN, H. J. **Discrete element simulations of dense packings and heaps made of spherical and non-spherical particles**. Powder technology, 109(1–3):278–292, 2000. 1

[31] MEFFERT, K. **Jgap: Java genetic algorithms package**. `http://jgap.sourceforge.net/`, Aug. 2009. 4.1.5

[32] MUNJIZA, A. **The combined finite–discrete element model**. 2004. 1

[33] MUNJIZA, A.; OWEN, D. R. J. ; BICANIC, N. **A combined finite element – discrete element modelling of rock blasting**. In: THIRD INTERNATIONAL CONFERENCE ON COMPUTATIONAL PLASTICITY: FUNDAMENTALS AND APPLICATIONS, p. 1903–1923, Barcelona, 1992. 1

[34] NAKAMURA, F. I. **Animação interativa de fluido baseada em partículas pelo método sph**. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, 2007. 1

[35] NASCIMENTO, E. L. D.; CARVALHO, JÚNIOR, H.; RAMOS, JÚNIOR, A. S. ; RAMOS, V. C. L. **Aplicação do método de elementos discretos na análise de estabilidade de taludes**. CMNE/CILAMCE, June 2007. 1

[36] NURKANOV, E. Y.; KADUSHNIKOV, R. M.; KAMENIN, I. G.; ALIEVSKII, D. M. ; KARTASHOV, V. V. **Investigation of the density characteristics of three-dimensional stochastic packs of spherical particles using a computer model**. Powder Metallurgy and Metal Ceramics, 40(5–6):229–235, May 2001. 2.3

[37] O'SULLIVAN, C. **The application of discrete element modelling to finite deformation problems in geomechanics**. Tese de Doutorado, University of California, Berkeley, 2002. 2.1

[38] POSE, M. G. **Introducción a los algoritmos genéticos**, 2008. 4.1.3, 4.1.4

[39] POTYONDY, D. O.; CUNDALL, P. A. ; SARRACINO, R. S. **Modeling of shock-and gas-driven fractures induced by a blast using bonded assemblies of spherical particles**, chapter 7, p. 55–62. Balkema, Rotterdam, 1996. 1, 1

[40] SAMET, H. **The design and analysis of spatial data structures**. Addison-Wesley Publishing, Reading, MA, 1989. 3.1, 3.1.1, 3.1.1, 3.1.1, 3.1.1

[41] SHI, G.-H. **Discontinuous deformation analysis: a new numerical model for the statics and dynamics of block systems**. Tese de Doutorado, University of California, Berkeley, nov 1988. 1

[42] SHIU, W.; DONZÉ, F. V. ; DAUDEVILLE, L. **Discrete element modelling of missile impacts on a reinforced concrete target**. In: INT. J. COMPUTER APPLICATIONS IN TECHNOLOGY, volumen 34, p. 33–41, 2009. 1

[43] SILVA, NETO, A. A. D. **Uma abordagem baseada em sph para animação interativa de águas rasas em jogos**. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, 2008. 1

[44] SPERBER, M. **Encyclopedia of GIS**, chapter Quadtree and Octree, p. 931–934. Springer, 2008. 3.1.1

[45] STROEVEN, P.; STROEVEN, M. **Dynamic computer simulation of concrete on different levels of the microstructure – part 1**. Image Anal Stereol, 22:91–95, 2003. 2.1

[46] VENETILLO, J. S. **Simulação de partículas baseada em gpu com tratamento de colisão**. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, 2007. 1

[47] VIEIRA, L. C. L. M. **Aplicação do método dos elementos discretos para análise de mecanismo de produção do material de sustentação de fraturas utilizado em poços de petróleo estimulados por fraturamento hidráulico**. Proposta de tese de Doutorado, nov 2007. 3

[48] WILLIAMS, J.; HOCKING, G. ; MUSTOE, G. **The theoretical basis of the discrete element method**. In: PROCEEDINGS OF THE NUMETA '85 CONFERENCE, p. 897–906, Swansea, Jan. 1985. 1

[49] ZHAO, D.; NEZAMI, E.; HASHASH, Y. M. A. ; GHABOUSSI, J. **Discrete element modeling of polyhedral representation of granular materials**. In: Malla, R. B.; Binienda, W. K. ; Maji, A. K., editors, EARTH & SPACE 2006: ENGINEERING, CONSTRUCTION, AND OPERATIONS IN CHALLENGING ENVIRONMENT, volumen 188, p. 77–77. ASCE, 2006. 1