MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

LUCIANE MACHADO FRAGA

PROPOSTA DE UM MÉTODO PARA OTIMIZAR DETECÇÃO DE COLISÕES UTILIZANDO ÁREAS DE INTERESSE

Rio de Janeiro
2009

INSTITUTO MILITAR DE ENGENHARIA

LUCIANE MACHADO FRAGA

PROPOSTA DE UM MÉTODO PARA OTIMIZAR DETECÇÃO DE COLISÕES UTILIZANDO ÁREAS DE INTERESSE

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Jauvane Cavalcante de Oliveira, Ph.D.
Co-orientador: Profa. Claudia Marcela Justel, D.Sc.

Rio de Janeiro
2009

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80-Praia Vermelha
Rio de Janeiro-RJ CEP 22290-270

F811d Fraga, Luciane Machado
Proposta de um Método para Otimizar Detecção de Colisões Utilizando Áreas de Interesse/ Luciane Machado Fraga.
– Rio de Janeiro: Instituto Militar de Engenharia, 2009.
118 p.: il., tab.

Dissertação (mestrado) – Instituto Militar de Engenharia – Rio de Janeiro, 2009.

1. Informática - teses 2. Descarte da Detecção de Colisão (Informática) 3. Ambientes Virtuais Dinâmicos I. Título. II. Instituto Militar de Engenharia.

CDD 003.3

INSTITUTO MILITAR DE ENGENHARIA

LUCIANE MACHADO FRAGA

PROPOSTA DE UM MÉTODO PARA OTIMIZAR DETECÇÃO DE COLISÕES UTILIZANDO ÁREAS DE INTERESSE

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Jauvane Cavalcante de Oliveira, Ph.D.

Aprovada em 29 de janeiro de 2009 pela seguinte Banca Examinadora:

Prof. Jauvane Cavalcante de Oliveira, Ph.D. do LNCC - Presidente

---

Profa. Claudia Marcela Justel, D.Sc. do IME

Prof. Paulo Renato da Costa Pereira - D.Sc. do IME

Prof. Selan Rodrigues dos Santos - Ph.D. da UFRN

Rio de Janeiro
2009

Dedico à minha família por todo apoio e força dados em mais essa conquista.

AGRADECIMENTOS

Agradeço primeiramente à minha família, em especial à Silvano Malfatti, pelo carinho e paciência.

Aos meus orientadores, pelo voto de confiança em mim depositado.

Aos meus grandes amigos, Cati e Selan, que foram os principais responsáveis pela conclusão desse trabalho e sempre se fizeram presentes nos momentos decisivos de minha vida. Não poderia deixar de agradecer também ao Loki, pelo enorme carinho que me recebeu em sua casa.

Aos meus amigos Caroline Barbosa e Henrique, Leandro Dihl, Cristiane Woszezenki, Teresa e Carlos dos Santos, Carla Pires, Ana Cristina Pletsch, Josiane Rodrigues e colegas do Laboratório ACiMA pelo incentivo na realização deste trabalho.

Por fim, a todos os professores e funcionários da Seção de Engenharia de Computação (SE/8) do Instituto Militar de Engenharia.

Luciane Machado Fraga

Porque esperar se podemos começar tudo de novo, agora mesmo. O sol nasce para todos, só não sabe quem não quer.

Legião Urbana

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SÍMBOLOS

ABREVIATURAS

AABB	-	*Axis Aligned Bounding Box*
AI	-	*Área de Interesse*
API	-	*Aplication Programming Interface*
AVC	-	*Ambiente Virtual Colaborativo*
BP	-	*Broad Phase*
BSP	-	*Binary Space Partitioning*
DOP	-	*Discrete-orientation Polytope*
FB	-	*Força Bruta*
EnCIMA	-	*Engine for Collaborative and Immersive Multimedia Applications*
GR	-	*Grades Regulares*
IME	-	*Instituto Militar de Engenharia*
MASSIVE	-	*Model, Architecture and System for Spatial Interaction in Virtual Environments*
NP	-	*Narrow Phase*
OBB	-	*Oriented Bounding Boxes*
OpenGL	-	*Open Graphics Library*
RV	-	*Realidade Virtual*
SaP	-	*Sweep and Prune*
VELVET	-	*Adaptive Hybrid Architecture for VEry Large Virtual EnvironmenTs*

RESUMO

Um dos principais desafios para aplicações de Realidade Virtual é a simulação em tempo real do comportamento realista de objetos em um ambiente virtual, principalmente no que se refere a detecção de colisão entre objetos em movimento. A quantidade de objetos envolvidos em uma simulação, bem como a complexidade geométrica dos objetos, está diretamente relacionada à quantidade de testes de interseção que devem ser realizados no processo de detecção de colisão. Isso faz com que no pior dos casos, testes sejam realizados entre todas as primitivas que compõem os objetos da cena, resultando em uma complexidade $O(n^2)$. Com a finalidade de otimizar o processo de detecção de colisão, a fase de descarte utiliza-se de métodos que identificam os pares de objetos que realmente podem colidir devido a estarem próximos no ambiente virtual. Desse modo, testes de interseção somente são realizados entre as primitivas dos pares de objetos que foram identificados pela fase de descarte. Devido a sua natureza estática, métodos utilizados na fase de descarte acabam não levando em consideração o caráter dinâmico dos ambientes virtuais, prejudicando deste modo o desempenho das aplicações devido a sobrecarga resultante das atualizações necessárias. Neste sentido, este trabalho estuda os principais métodos utilizados na fase de descarte e apresenta o método *AoIP* (*Area of Interest Partitioning*) para a redução do número de objetos a serem considerados nos testes de colisão em ambientes virtuais.

ABSTRACT

One of the main challenges for Virtual Reality applications is the real time simulation of the correct behaviour of objects in the virtual world, special attention must be given to collision detection in order to maintain the expected behaviour. The ammount of 3D objects in a simulation, as well as the inherent geometric complexity, isproportional to the ammount of intersection tests needed to be performed aiming at detecting collisions. In the worst case scenario, all primitive components of each object must be compared with all others, which leads to an $O(n^2)$ complexity algorithm. In order to optimize the collision detection process, the *broad phase* uses methods that aim at identifying pairs of objects whose probability of collision is very high, considering their proximity in the Virtual Reality world. Once identified, only primitives from such objects are tested against each other, reducing significantly the complexity of the algorithm. Most *Broad Phase* methods do not handle the dinamism of Virtual Reality worlds due to their static nature. Those methods lead to some complexity added due to the required updates needed by such methods. This work studies most Broad Phase methods and introduce a novel method, Area of Interest Partitioning (AoIP), that reduce the ammount of objects that need to be considered for the detailed collision detection phase that follows in Virtual Reality worlds.

1 INTRODUÇÃO

A Realidade Virtual (RV) é vista como uma tecnologia amplamente utilizada no apoio a educação e treinamento em áreas como Psicologia, Medicina, Treinamento Militar, Engenharia, dentre outras. Segundo Kim (KIM, 2005), isso se deve ao fato de permitir o desenvolvimento de ambientes virtuais que apresentem imersão e interatividade em tempo real (KIM, 2005), sendo a combinação desses 2 fatores o responsável pelo realismo em diferentes aplicações.

Os sistemas de Realidade Virtual permitem criar ambientes onde os usuários podem visualizar, explorar e interagir com objetos virtuais. Com isso, surge a necessidade de simular via computador, certas propriedades de maneira tal que o ambiente reaja da forma esperada. Por exemplo, em um ambiente virtual objetos não devem atravessar uns aos outros e devem mover-se da forma esperada quando são empurrados, puxados ou agarrados. Para poder simular um ambiente interativo respeitando estas restrições torna-se necessário um mecanismo de detecção e reação a colisão.

A quantidade de testes de intersecção que devem ser realizados no processo de detecção de colisão está diretamente relacionada a quantidade de objetos envolvidos em uma simulação, bem como a complexidade geométrica desses objetos. Isso faz com que, no pior dos casos, testes de intersecção sejam realizados entre todos os polígonos que compõem os objetos de uma cena, resultando em uma complexidade temporal $O(n^2)$.

Para contornar este problema o processo de detecção de colisão pode ser tratado em duas fases: fase de descarte (*broad phase*) e fase de refinamento (*narrow phase*) (ERICSON, 2005). Realizar testes de intersecção entre todos os objetos de uma cena torna-se desnecessário, principalmente quando se verifica que muitos destes objetos encontram-se distantes uns dos outros. Com isso, a proximidade dos objetos pode ser um fator levado em consideração para a redução dos testes de intersecção a serem realizados.

Com a finalidade de otimizar o processo de detecção de colisão, a fase de descarte utiliza-se de métodos que identificam os pares de objetos que realmente podem colidir devido a estarem próximos no ambiente virtual. Desse modo, testes de intersecção somente são realizados na fase de refinamento entre as primitivas dos pares de objetos que foram identificados pela fase de descarte.

1.1 MOTIVAÇÃO

Um dos principais desafios para aplicações de Realidade Virtual é a simulação em tempo real do comportamento realista de objetos em um ambiente virtual, principalmente no que se refere a detecção de colisão entre objetos em movimento. Reduzir a quantidade de objetos a serem testados no processo de detecção de colisão, pode reduzir o custo computacional inerente a este processo.

Devido a sua natureza estática, métodos utilizados na fase de descarte acabam não levando em consideração o caráter dinâmico dos ambientes virtuais, prejudicando deste modo o desempenho das aplicações devido a sobrecarga resultante das atualizações necessárias.

Os problemas enfrentados no processo de detecção de colisão são muito semelhantes aos problemas encontrados em Ambientes Virtuais Colaborativos (AVCs), que são ambientes que permitem a interação entre usuários distribuídos geograficamente com o objetivo de realizarem alguma tarefa colaborativamente. Sem a utilização de um método adequado para a redução de mensagens trocadas entre usuários em um AVC, quanto maior o número de objetos no ambiente maior será a troca de mensagens através da rede. No caso da detecção de colisão, quanto maior o número de objetos no ambiente maior será a quantidade de pares de objetos a serem testados.

Em AVCs, o conceito de áreas de interesse tem sido utilizado como uma esquema de filtragem para a redução da troca de mensagens (OLIVEIRA, 2003). Como este esquema tem se mostrado eficiente para AVCs, decidiu-se adaptar esta idéia a ambientes virtuais na fase de descarte, para otimizar o processo de detecção de colisão.

1.2 OBJETIVOS

O objetivo principal deste trabalho é aplicar o conceito de áreas de interesse como um mecanismo para reduzir o número de objetos a serem considerados nos testes para detecção de colisão em ambientes virtuais, de modo a acelerar a fase de refinamento.

Para atingir este objetivo foram realizadas as seguintes etapas:

- Caracterização dos métodos de descarte para ambientes virtuais;
- Caracterização dos métodos de gerenciamento de áreas de interesse;

- Proposta e implementação de uma solução para o problema mencionado utilizando o conceito de áreas de interesse;
- Realização de experimentos para validar a solução proposta através de um ambiente de simulação.
- Análise dos resultados obtidos.

1.3 ESTRUTURA DA DISSERTAÇÃO

Esta dissertação está organizada da seguinte maneira: no Capítulo 2 é apresentado o processo de detecção de colisão, bem como alguns dos fatores que influenciam este processo; no Capítulo 3 são descritos os métodos de detecção de colisão existentes utilizados nas fase de descarte; no Capítulo 4 o conceito de áreas de interesse é definido e o Método de Descarte proposto é apresentado; no Capítulo 5 o ambiente de simulação é definido; no Capítulo 6 são apresentados os experimentos realizados e a análise dos resultados obtidos; no Capítulo 7 são realizadas as considerações finais sobre o trabalho desenvolvido e trabalhos futuros são sugeridos.

2 TRATAMENTO DE COLISÃO

O tratamento de colisão entre objetos tem sido investigado devido a sua relevância para o desenvolvimento de aplicações gráficas interativas em áreas como robótica, automação e simulação de ambientes virtuais (LIN, 1998).

Dois processos estão envolvidos no tratamento de colisão: o processo de detecção de colisão e o processo de resposta à colisão (MOORE, 1988). O processo de detecção de colisão consiste em verificar se dois ou mais objetos estão se intersectando. O processo de resposta à colisão consiste em determinar quais são as ações que devem ser realizadas quando uma colisão é indentificada entre os objetos.

Este capítulo apresenta o processo de detecção de colisão e alguns dos fatores que influenciam este processo.

2.1 O PROCESSO DE DETECÇÃO DE COLISÃO

Detectar uma colisão é verificar o momento em que ocorre a intersecção entre dois objetos (BERGEN, 2004). Com isso, a detecção de colisão tem a finalidade de identificar quando objetos virtuais poderão se sobrepor para que uma reação correta seja gerada pelo ambiente, fornecendo assim maior realismo às aplicações.

A detecção de colisão ainda é um dos grandes gargalos para o desenvolvimento de aplicações interativas devido a este processo consumir grande parte do processamento, fazendo com que as taxas de atualização sejam reduzidas e, consequentemente, o desempenho da aplicação seja prejudicado (ROCHA, 2007). De acordo com Fares e Haman (FARES, 2005), esse problema pode ser influenciado pelos seguintes fatores:

- Representação dos objetos: há muitas maneiras de representar objetos em uma cena, seja de forma implícita ou explícita. Quando a representação é definida em termos de vértices, arestas e faces, ela é dita explícita. Já quando a representação é definida por expressões matemáticas ela é dita implícita. Esferas, cones e cilindros são primitivas definidas implicitamente (LIN, 1998).

  Modelos poligonais são as representações mais utilizadas em computação gráfica e modelagem, por serem bastante simples. Polígonos podem ser conectados uns

aos outros através de suas arestas para formar uma extensa superfície poligonal, conhecida como malha (MOLLER, 1971). Geralmente a representação utilizada em muitos ambientes virtuais para os objetos são malhas de triângulos.

Objetos geométricos podem ser de dois tipos: convexos ou côncavos. Um objeto é convexo se contém todos os segmentos de reta que conectam pares de pontos, caso contrário é chamado côncavo. Em geral, os objetos convexos permitem executar verificações de intersecção mais rapidamente do que os objetos côncavos (BERGEN, 2004).

A representação geométrica usada para uma cena e seus objetos tem influência direta sobre os métodos a serem utilizados para a detecção de colisão. Quanto mais simples forem as representações adotadas, maiores são as chances de serem encontradas soluções mais genéricas que possibilitem a redução do custo computacional para o problema de detecção de colisão (LIN, 1998).

- Tipos de Testes de Colisão: diferentes aplicações necessitam de diferentes testes de colisão. Geralmente quanto há necessidade de resultados mais precisos sobre a ocorrência de uma colisão, mais detalhados e complexos são os testes a serem realizados e maior é o custo computacional envolvido.

  O teste de colisão a ser utilizado entre dois objetos é o teste de intersecção, o qual retorna como resultado um valor lógico indicando se dois objetos estão ou não colidindo. Ás vezes não é suficiente saber somente se os objetos estão colidindo, mas também é necessário determinar quais partes dos objetos estão se intersectando. Determinar o conjunto de pontos de contato não é uma tarefa trivial (ERICSON, 2005).

  Em algumas aplicações como jogos, testes de intersecção aproximados são suficientes. No entanto, para aplicações como simuladores médicos, a precisão na determinação dos pontos de contato é de fundamental importância.

  Outro tipo de teste a ser considerado é aquele que determina o valor correspondente a profundidade da intersecção entre dois objetos. Algumas aplicações necessitam dessa informação, principalmente as que trabalham com objetos deformáveis e renderização de força.

- Características do Ambiente de Simulação: características específicas de cada simu-

lação são frequentemente consideradas para a escolha apropriada de métodos de detecção de colisão. Tais características referem-se às propriedades dos objetos de acordo com sua natureza e tipos de movimentos que podem realizar, e à quantidade de objetos presentes em um ambiente virtual.

Um ambiente 3D é um mundo virtual composto por diversos objetos que dependendo de sua natureza e da finalidade da aplicação, possuem formas e tamanhos variados, podem ser estáticos ou dinâmicos, rígidos ou deformáveis (KIM, 2005).

Considera-se que um objeto é estático quando não possui movimento e que ele é dinâmico quando, com o passar do tempo, movimenta-se pelo ambiente, seja com uma velocidade constante ou não. O movimento de um objeto é o resultado da mudança na sua posição e/ou orientação.

Objetos rígidos são aqueles em que somente a sua posição e orientação podem ser alteradas. Rotações e translações são transformações geométricas aplicadas a estes tipos de objetos de modo que sejam movimentados sem qualquer deformação. Já objetos deformáveis, além de poderem ter sua posição e orientação alteradas, podem apresentar deformações no decorrer do tempo como por exemplo, resultado de uma reação do contato exercido sobre eles.

Em ambientes virtuais, a presença de objetos que apresentem movimentos aumenta o realismo da aplicação. Estes tipos de ambientes enquanto visualizados em tempo real por um observador que se movimenta arbitrariamente pelo espaço, permitem que seus objetos se movimentem também de forma arbitrária e possibilitam inserções e exclusões de objetos a qualquer momento. Ao contrário, em um ambiente virtual estático, apenas o observador se move enquanto a cena é exibida em tempo real.

Além das propriedades, a quantidade de objetos envolvidos em uma simulação também é um fator crítico para o processo de detecção de colisão. Quanto mais objetos forem inseridos em um ambiente maior será a quantidade de testes de colisão a serem realizados.

O número de objetos envolvidos em uma simulação e a complexidade dos mesmos determinam a quantidade de testes que devem ser realizados para a detecção de colisão. Isso faz com que, no pior dos casos, testes sejam realizados entre todas as primitivas que compõem os objetos da cena, resultando em uma complexidade $O(n^2)$ (BERGEN, 2004). Assim, para n objetos haveriam $(\frac{n}{2}) = (n(n-1))/2$ pares de objetos potencialmente

colidíveis. Obviamente, este método conhecido como *Força Bruta*, se torna inviável para ambientes com um número muito grande de objetos e que requer simulações em tempo real.

Volumes envoltórios podem ser uma alternativa para acelerar a verificação de colisões, visto que abstraem a representação geométrica dos objetos em questão (JIMÉNEZ, 2001). Alguns dos tipos de volumes envoltórios encontrados na literatura são apresentados a seguir.

2.1.1 VOLUMES ENVOLTÓRIOS

A idéia de utilizar volumes consiste em encapsular cada objeto da cena de maneira que a ocorrência ou não de uma colisão seja obtida através de testes de intersecção entre estes volumes (FARES, 2005).

Testes de intersecção entre volumes de dois objetos são muito mais rápidos do que testes de intersecção realizados entre todos os seus polígonos. Ao se constatar que dois volumes envoltórios não estão colidindo, assume-se que todos os polígonos que compõem uma objeto também não estão.

Algumas propriedades são desejáveis para os volumes envoltórios utilizados (BERGEN, 2004):

- Possuir uma forma geométrica simples e que se ajuste de forma mais precisa possível a geometria do objeto;
- Apresentar testes de colisão computacionalmente simples;
- Possibilitar um rápido cálculo do volume;
- Requerer pouca memória para armazenamento.

Geralmente os volumes utilizados são primitivas geométricas, como esferas e caixas, devido à simplicidade encontrada na construção e realização de testes de intersecção com estas primitivas. Em muitos casos estes volumes não se ajustam perfeitamente a certos tipos de objetos, principalmente aqueles não simétricos, o que pode ocasionar uma falsa colisão. Existem volumes mais complexos que são utilizados com o objetivo de ter-se um ajuste mais refinado e evitar falsas colisões. No entanto, geralmente com estes volumes o baixo custo computacional para computação e testes de intersecções é comprometido (LIN, 1998).

Uma solução para o alto custo de computação de certos volumes seria computá-los em uma fase de pré-processamento, ao invés de em tempo de execução (ERICSON, 2005). Porém, para algumas aplicações é necessário que um volume seja reconstruído constantemente devido a transformações realizadas sobre o objeto, como por exemplo quando utilizados objetos deformáveis. Nestes casos, haverá vezes em que seja mais vantajoso computar um novo volume do que reconstruí-lo.

A quantidade de memória necessária para o armazenamento de um volume está relacionada à simplicidade do tipo de volume escolhido. Formas geométricas simples exigem estruturas de dados menos complexas e menos espaço de memória para armazená-las.

A Figura 2.1 apresenta os tipos de volumes envoltórios mais utilizados para a detecção de colisão, que serão descritos a seguir.

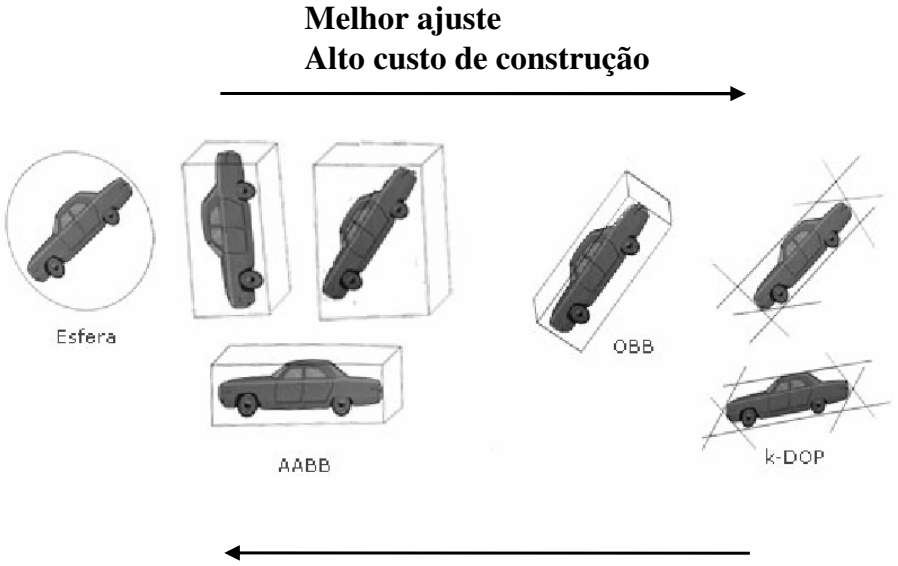


FIG. 2.1: Tipos de volumes envoltórios organizados pela complexidade de construção, custo computacional dos testes de colisão e ajuste ao objeto.

2.1.1.1 ENVOLTÓRIOS POR ESFERAS

Também encontrado na literatura como *BoundingSpheres*, o uso de esferas como volume envoltório é muito comum devido sua simplicidade de construção, sendo que para seu armazenamento somente são necessárias as coordenadas do seu centro e o tamanho do seu raio.

Apesar de esferas nem sempre se ajustarem adequadamente a um objeto, este tipo de volume é muito utilizado em ambientes virtuais devido aos seguintes fatores:

- Simplicidade de atualização;
- Testes de intersecção simples sendo necessárias poucas operações aritméticas;
- Invariantes à rotações, não sendo restritas a uma orientação específica, tendo que apenas serem transladadas para a sua nova posição.

2.1.1.2 ENVOLTÓRIOS POR CAIXAS ALINHADAS AOS EIXOS

Conhecidos como *AABBs* (*Axis Aligned Bounding Boxes*), pode-se dizer que estes envoltórios são utilizados em igual ou maior proporção que esferas. Assim como esferas, AABBs são fáceis de computar e permitem testes de intersecções com um número pequeno de operações (ERICSON, 2005).

Apesar de precisarem de mais espaço de armazenamento que as esferas, *AABBs* ajustam-se melhor a determinados objetos, sendo necessário para seu armazenamento um conjunto de coordenadas referentes aos pontos máximo e mínimo ortogonalmente projetados sobre os eixos X, Y e Z. Para testes de intersecção apenas é necessária a comparação direta entre valores individuais de coordenadas.

Devido a *AABBs* serem restritos à orientação que eles podem assumir, a simples rotação de um objeto pode fazer com que o volume tenha de ser inteiramente reconstruído para se a justar às novas posições do objeto. Para evitar o processo de reconstrução do volume em situações como essa, *AABBs* podem ser definidas com um tamanho fixo, de tal forma que seu tamanho seja determinado a partir do tamanho máximo e mínimo do objeto referentes a todas as possíveis orientações.

Em casos onde objetos apresentam largura e altura desproporcionais essa solução acaba gerando caixas muito maiores que os objetos, e em conseqüência, colisões que não estão ocorrendo poderão ser detectadas. *AABBs* de tamanho variável não sofrem deste problema, visto que seu volume é recalculado constantemente para se ajustar a rotação do objeto durante seu movimento.

2.1.1.3 ENVOLTÓRIOS POR POLITOPOS DE ORIENTAÇÃO DISCRETA

Envoltórios por Politopos de Orientação Discreta são conhecidos como *K-DOPs* (*Discrete-orientation Polytopes*). Estes tipos de volumes são uma generalização das *AABBs*, onde

as características definidas para *AABBs* são também aplicadas a *k-DOPs*. No entanto, *k-DOPs* permitem a definição de volumes mais ajustáveis usando k eixos, ao invés de apenas os eixos coordenados utilizados pelas *AABBS* (ERICSON, 2005). *AABBs* são considerados casos especiais de 6-DOPs, devido às faces de uma caixa estarem alinhadas às 6 direções dos 3 eixos coordenados.

K-DOPs são entidades geométricas definidas a partir das direções normais das faces do objeto, dos pontos extremos (mínimos e máximos) desse objeto e de $k/2$ planos paralelos com direções pré-definidas. A partir dessas informações é possível a definição de todas as faces que irão compor o *K-DOP*. As direções das faces dos *K-DOPs* são fixas, mas não necessariamente coincidem com os eixos coordenados, podendo haver mais que três direções para as faces.

2.1.1.4 ENVOLTÓRIOS POR CAIXAS ORIENTADAS

Caixas orientadas ou *OBBs* (*Oriented Bounding Boxes*) não são os volumes mais econômicos em termos de quantidade de espaço de armazenamento e custo computacional para os testes de intersecção. Porém, este tipo de volume é o que mais se aproxima à superfície dos objetos por ter uma orientação arbitrária (ERICSON, 2005).

Para garantir um ajuste quase que perfeito sobre um objeto, *OBBs* utilizam uma matriz $3X3$ para representar sua orientação, devido a tais estruturas proporcionarem testes de intersecção mais eficientes (FARES, 2005).

A construção de um *OBB* para um objeto pode ser realizada a partir da determinação dos 3 eixos ortogonais que constituem as orientações das faces dos objetos (BERGEN, 2004).

2.1.2 CONSIDERAÇÕES

Apesar de testes de intersecção entre volumes serem muito mais rápidos do que testes de intersecção realizados entre os polígonos que compõem a malha correspondente de um objeto, a utilização deste método ainda requer que todos os objetos sejam comparados entre si. Para evitar este problema, é preciso adotar medidas que reduzam a quantidade de objetos a serem tratados. A seção 2.2 apresenta como o processo de detecção de colisão pode ser otimizado.

2.2 FASES DO PROCESSO DE DETECÇÃO DE COLISÃO

A evolução do hardware gráfico possibilita que cada vez mais polígonos possam ser renderizados em tempo real. No entanto, mesmo para os processadores mais modernos realizar testes de colisão com uma grande quantidade de polígonos ainda é um problema, o que exige a implementação de métodos mais eficientes e que busquem a redução de cálculos.

Vários métodos têm sido propostos para a detecção de colisão com a finalidade de aumentar o desempenho das aplicações. Segundo Ericson (ERICSON, 2005), estes métodos geralmente resultam em uma abordagem de detecção de colisão híbrida, ou seja, que possibilita resolver o problema em duas fases:

- Fase de descarte: esta fase, apesar de não envolver testes entre polígonos, ainda pode ser custosa computacionalmente se forem realizados testes de colisão entre todos os objetos presentes no ambiente virtual (método da *Força Bruta*).

  Para evitar a comparação de todos os pares de objetos em um ambiente virtual, torna-se necessário descartar objetos que tem pouca probabilidade de colisão em um determinado momento.

  A fase de descarte utiliza métodos que identificam pequenos grupos de objetos que podem estar colidindo, utilizando conceitos como a coerência espacial e/ou temporal (BERGEN, 2004). Na coerência espacial considera-se que um objeto ocupa uma região do espaço. Para a coêrencia temporal um objeto tende a ocupar a mesma região do espaço de um instante de tempo para outro.

- Fase de Refinamento: esta fase utiliza métodos que atuam sobre os polígonos que formam a geometria dos objetos identificados pela fase de descarte. Portanto, o refinamento determina com mais precisão onde as colisões ocorreram.

O métodos de descarte utilizam abordagens mais simples as quais não fornecem altos níveis de precisão. Tais abordagens baseiam-se em rejeitar objetos de acordo com a região do espaço que eles ocupam. O Capítulo 3 apresenta os métodos que podem ser utilizados para a redução de objetos na fase de descarte.

3 FASE DE DESCARTE DO PROCESSO DE DETECÇÃO DE COLISÃO

A fase de descarte tem importante papel no processo de detecção de colisão. É a partir do descarte de objetos que não estão próximos o bastante para colidirem que o processo de detecção torna-se mais rápido, e consequentemente o desempenho da aplicação é melhorado.

Métodos para a fase de descarte têm sido propostos para otimizar o processo de detecção de colisão em ambientes tridimensionais. Podemos classificar estes métodos em duas categorias: métodos baseados no particionamento espacial e métodos baseados na varredura do espaço. Na primeira categoria encontram-se métodos que dividem o espaço e então comparam os diferentes objetos somente se estes se encontram na mesma partição espacial. A segunda categoria contém métodos que realizam a varredura espacial do espaço em busca de objetos que tem suas projeções em sobreposição.

Neste capítulo são caracterizados alguns dos métodos de descarte existentes para cada categoria e analisados seus pontos positivos e negativos no que se refere à sua aplicação para ambientes virtuais dinâmicos.

3.1 MÉTODOS DE DESCARTE BASEADOS NO PARTICIONAMENTO ESPACIAL

O particionamento espacial consiste em subdividir o espaço em regiões (SAMET, 1990). Cada partição mantém uma lista de referências a objetos que nela estão totalmente ou parcialmente contidos, e, com isso, apenas são realizados testes para objetos pertencentes a uma mesma região. Neste contexto, a coerência espacial passa a ser utilizada como uma alternativa de otimização para a fase de descarte, onde pares de objetos podem ser descartados rapidamente dos testes de colisão em função da região do espaço que eles ocupam.

É possível particionar um ambiente em um grande número de regiões, diminuindo consideravelmente o número de testes de colisão a serem realizados. No entanto, o processamento exigido pela atualização das estruturas neste caso pode comprometer o desempenho do processo de detecção de colisão como um todo. Para reduzir a complexidade da manutenção das estruturas, pode-se aproveitar o conceito de coerência temporal. Este conceito leva em consideração a seguinte premissa: se um par de objetos colidiu em um

determinado instante (e ambos ainda existem) é provável que continuem colidindo posteriormente.

Métodos de particionamento espacial foram desenvolvidos para diminuir o número de pares de objetos a serem testados no processo de detecção de colisão. Estes métodos se baseiam na identificação de objetos que podem colidir por se encontrarem próximos no ambiente virtual. Tais métodos podem ser classificados como não-hierárquicos e hierárquicos (AGARWAL, 2008).

Entre os métodos com particionamento espacial não-hierárquico está a *Grade Regular* (BERGEN, 2004; ERICSON, 2005), a qual divide o espaço em um conjunto de células de tamanho uniforme em cada eixo xoordenado. *Octrees* (SAMET, 1990), *Kd-Trees* (BENTLEY, 1975) e *BSP-Trees* (FUCHS, 1980) são alguns dos métodos com particionamento espacial hierárquico, os quais dividem o espaço em várias regiões. As regiões criadas são recursivamente divididas em novas regiões conforme o critério de particionamento adotado por cada método. As regiões e sub-regiões obtidas por um método com particionamento espacial hierárquico são organizadas na forma de uma árvore. Os Métodos não-hierárquicos e hierárquicos supracitados serão descritos a seguir.

3.1.1 *GRADES REGULARES*

Grades Regulares (ERICSON, 2005) particionam o espaço em células cúbicas de mesmas dimensões, também chamadas de *voxels*. O tamanho das células de uma *Grade Regular* é baseado em uma granularidade pré-definida. Para células de tamanho $N1$ x $N2$ x $N3$, $N1$, $N2$ e $N3$ correspondem ao número de subdivisões a serem realizadas para os eixos X, Y e Z, respectivamente.

Para cada célula de uma *Grade Regular* é mantida uma lista de objetos (ou partes de objetos) que intersectam a célula. Desse modo, o método *Grades Regulares* busca diminuir o número de testes da fase de descarte através da identificação dos objetos que se encontram em uma mesma célula. Com isto, apenas os objetos pertencentes a uma mesma lista serão considerados para os testes de colisão.

Pode-se dizer que as células definidas para uma *Grade Regular* formam uma estrutura estática, visto que não se alteram após serem criadas. Somente as listas de objetos de cada célula são atualizadas, de modo a armazenarem de forma consistente os objetos que se encontram em uma determinada célula num instante de tempo específico. A Figura 3.1 mostra uma representação bidimensional de uma *Grade Regular* e o mapeamento dos

objetos para as respectivas listas de cada célula. Os objetos que intersectam mais de uma célula são duplicados.

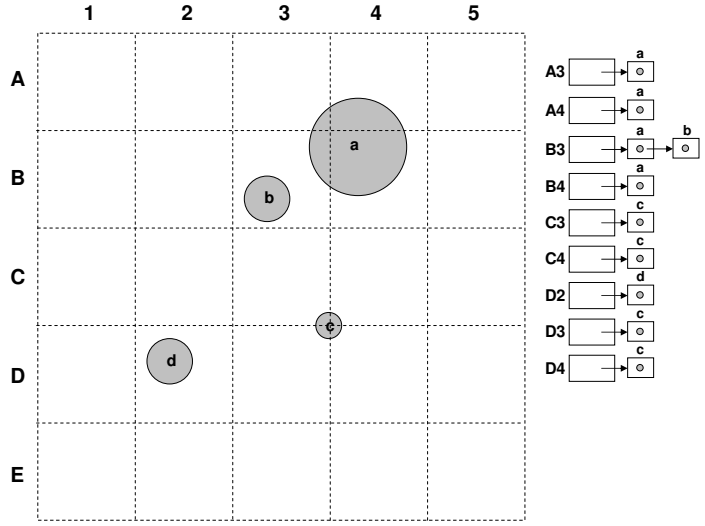


FIG. 3.1: Representação bidimensional de uma *Grade Regular* 5X5 com 4 objetos inseridos no ambiente.

Devido a sua natureza estática e uniforme, acessar uma célula torna-se uma operação simples, assim como é a obtenção da vizinhança de uma célula específica (BERGEN, 2004). Apesar de ser simples para construir e percorrer, *Grades Regulares* geram células mesmo para partes do ambiente virtual onde não existem objetos, tornando a pesquisa mais custosa e a utilização de memória pouco eficiente. Além disso, o desempenho obtido por uma *Grade Regular* depende principalmente dos seguintes fatores:

a) da escolha de um tamanho adequado das células;

b) da densidade de ocupação do espaço, e;

c) do tamanho dos objetos encontrados no ambiente virtual.

Para diferentes cenários é dificil definir a melhor granularidade de uma *Grade Regular*. Se o tamanho das células é definido como muito pequeno, pode ocorrer de um grande número de células necessitarem ser atualizadas, ainda mais para ambientes virtuais dinâmicos. Ainda para um tamanho de célula pequeno, se o ambiente contiver objetos muito grandes, haverá neste caso a duplicação de muitos objetos, visto que um mesmo objeto estará intersectando várias células resultando em um alto consumo de memória

para armazenamento. Para células definidas com um tamanho muito grande, se os objetos do ambiente forem muito pequenos, haverá muitos objetos em uma mesma célula, o que pode levar ao pior caso se os objetos estiverem aglomerados.

Cenários com aglomeração de objetos geralmente são comuns em muitas das aplicações em Realidade Virtual (RV). Nestes ambientes pode ocorrer de somente algumas poucas células conterem uma quantidade muito grande de objetos enquanto que a maioria das células estarão vazias. Neste caso, *Grades Regulares* passam a não ser tão eficientes na rejeição de pares de objetos para os testes de intersecção. Deste modo, *Grades Regulares* são mais indicadas para ambientes virtuais onde os objetos estejam uniformemente distribuídos.

3.1.2 *QUADTREES E OCTREES*

Quadtrees e *Octrees* subdividem o espaço 3D hierarquicamente em células retangulares, onde cada nó na árvore corresponde a uma região do espaço. *Quadtrees* são estruturas aplicadas ao espaço 2D e os nós internos da árvore representam regiões do espaço que são subdivididas recursivamente em quatro partições de mesmas dimensões alinhadas aos eixos, chamadas quadrantes. Já *Octrees* são adaptadas para o espaço 3D onde os nós internos são regiões que são divididas recursivamente em oito partições de mesmas dimensões alinhadas aos eixos, chamadas octantes (SAMET, 1990). Para ambos métodos a recursão não é finalizada até que um determinado critério de parada seja satisfeito. Geralmente os critérios utilizados são até que uma profundidade máxima seja alcançada ou ainda até que uma quantidade específica de objetos por nó seja encontrada.

Como o próprio nome sugere, cada nó em uma *Quadtree* tem 4 filhos e em uma *Octree* oito filhos. Nós internos podem ter como filhos outros nós internos ou nós folhas que correspondem aos objetos (ou partes dos objetos) contidos no espaço (ERICSON, 2005). A Figura 3.2 ilustra a divisão do espaço bidimensional por uma *Quadtree* e a Figura 3.3 ilustra a divisão do espaço tridimensional por uma *Octree*. Nestes exemplos, objetos que intersectam mais de uma região são duplicados.

Na Figura 3.3 o nó raiz da árvore, representado por uma elipse, corresponde ao ambiente como um todo e contém 8 filhos. As folhas de cada nível, representadas por um quadrado branco, indicam octantes com nenhum objeto. Octantes que contém um ou mais objetos são representadas por quadrados cinzas. Neste exemplo a condição de parada de recursão utilizada foi a profundidade pré-definida por 2, onde cada nó folha no

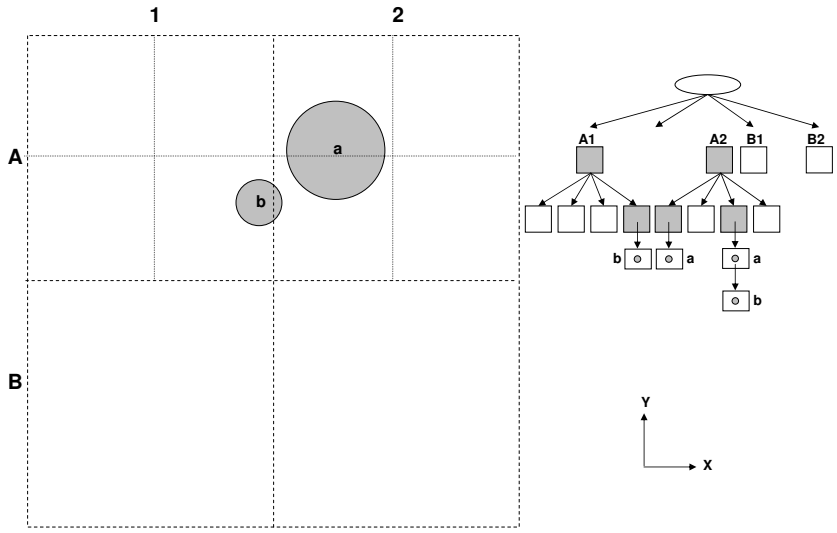


FIG. 3.2: Divisão do espaço por uma *Quadtree* a partir de 2 objetos inseridos no ambiente.

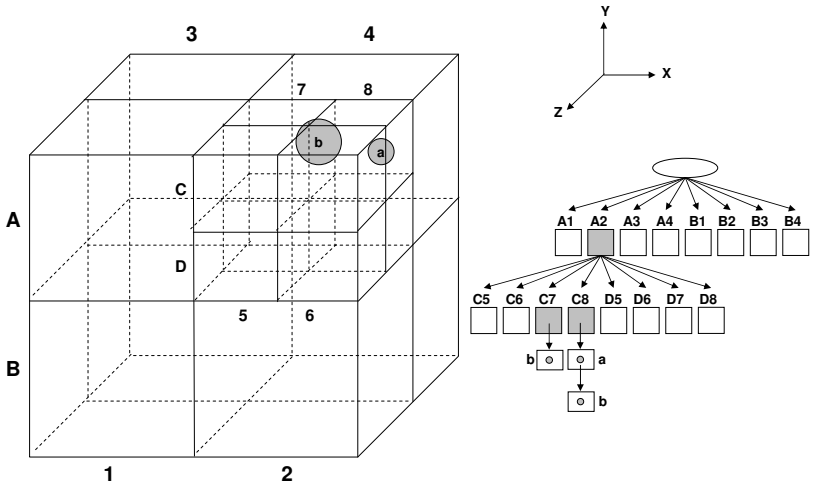


FIG. 3.3: Divisão do espaço por uma *Octree* a partir de 2 objetos inseridos no ambiente.

nível 2 contém a lista de possíveis candidatos à colisão.

Para uma *Octree* é dificil estimar a profundidade quando aplicada em diferentes ambientes virtuais. Quanto maior a profundidade adotada menores serão as regiões obtidas visto que mais subdivisões serão realizadas. Isto implica em uma quantidade significativa de atualizações, principalmente para ambientes dinâmicos, onde uma *Octree* necessita ser reconstruída a cada movimento dos objetos, visto que não há como garantir que as listas de cada célula continuarão válidas após sua criação.

3.1.3 *KD-TREES*

Kd-Trees são estruturas hierárquicas de particionamento do espaço multidimensional, sendo representadas por uma árvore binária k-dimensional. As *Kd-Trees* dividem um ambiente virtual em regiões de acordo com planos ortogonais alinhados a uma sequência de eixos pré-definidos do sistema de coordenadas utilizado. Cada região não necessita ter as mesmas dimensões.

O eixo a que um plano de divisão deve estar alinhado pode ser selecionado entre as k dimensões suportadas pelo sistema de coordenadas utilizado. Além da seleção do melhor plano de divisão, o posicionamento deste plano de divisão deve ser definido. Por exemplo, as *Kd-Trees* podem particionar o espaço tendo os planos de divisão alinhados ao eixo X, depois ao eixo Y, depois ao eixo Z, então aos eixos X, Y e Z novamente e assim por diante.

Um critério de parada da recursão também deve ser especificado para *Kd-Trees*. Outra questão que deve ser levada em consideração em tais estruturas é como classificar os objetos com relação a esse plano. Sugere-se que objetos que estão dispostos em mais do que uma região sejam duplicados para cada região ou sua geometria decomposta. Tais alternativas podem funcionar bem para objetos estáticos, no entanto para objetos dinâmicos elas se tornam custosas computacionalmente devido às atualizações necessárias a cada vez que o objeto se move.

A divisão do espaço em regiões menores torna as *Kd-Trees* estruturas mais flexíveis, visto que podem se adaptar mais facilmente a objetos de formatos variados.

A Figura 3.4 mostra uma representação bidimensional de uma *Kd-Tree* a qual alterna entre os eixos X e Y a cada nível da recursão para alinhar o plano de partição. Uma posição arbitrária é escolhida para cada plano de divisão. O critério de parada de recursão utilizado foi a profundidade máxima da árvore de 2. Os objetos que se encontram em um

mesmo nó folha serão considerados para os testes de colisão.

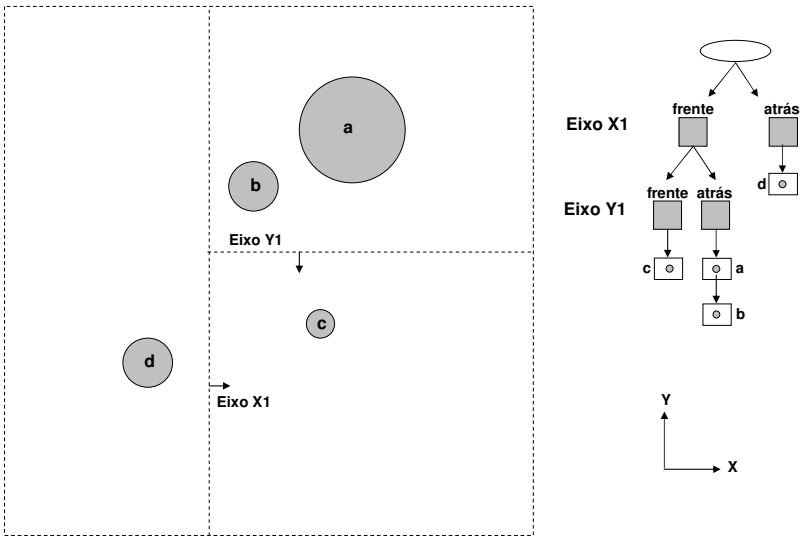


FIG. 3.4: Representação bidimensional de uma *Kd-Tree* com quatro objetos inseridos no ambiente.

3.1.4 *BSP-TREES*

BSP-Trees foram originalmente desenvolvidas para resolver o problema das superfícies escondidas de maneira a descartar rapidamente objetos que não necessitam ser renderizados (FUCHS, 1980).

Uma *BSP-Tree* é uma árvore binária que divide o espaço recursivamente por meio de um plano de divisão de orientação e posição arbitrárias. Ao contrário da *Grade Regular*, uma *BSP-Tree* permite divisões não uniformes, o que possibilita uma melhor adaptação à distribuição espacial dos objetos (LUQUE, 2005).

A Figura 3.5 mostra uma representação bidimensional de uma *BSP-Tree*. No exemplo, a condição de parada para a recursão adotada foi a profundidade máxima de 2 para a árvore. Objetos localizados em um mesmo nó folha são considerados para os testes de colisão.

O fato da escolha do plano de divisão de uma *BSP-Tree* ter orientação e posição arbitrários faz com que a divisão em regiões seja mais eficiente. No entanto, essa escolha não é uma tarefa complexa. Segundo Luque et al (LUQUE, 2005), planos de divisão podem ser escolhidos utilizando as seguintes métricas :

- população: número de objetos classificados em relação ao plano;

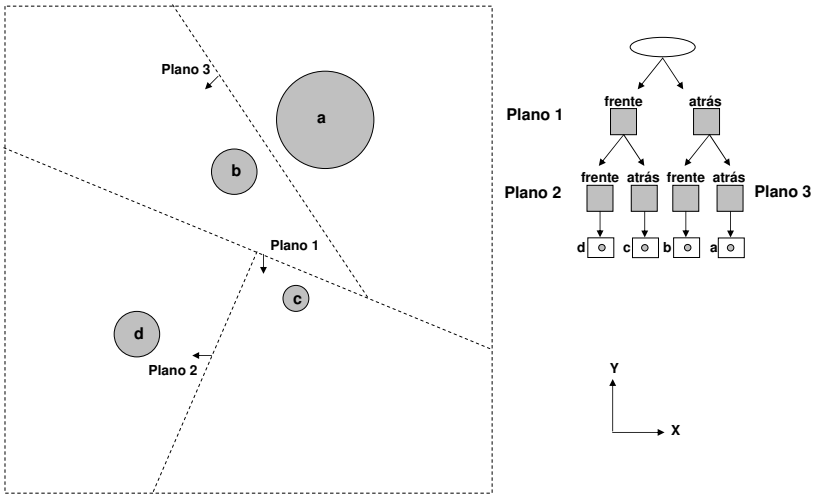


FIG. 3.5: Representação bidimensional de uma *BSP-Tree* com 4 objetos inseridos no ambiente.

- balanceamento: como os objetos estão distribuídos nas sub-árvores de um nó, ou seja, a diferença entre a quantidade de objetos que estão na frente e atrás do plano de divisão deve ser próxima de zero;
- redundância: quantidande de objetos que o plano intersecta e que deverão ser duplicados nas sub-árvores.

3.2 MÉTODO DE DESCARTE BASEADO NA VARREDURA ESPACIAL

Métodos de varredura espacial reduzem o número de pares de objetos a serem testados no processo de detecção de colisão através da busca no ambiente virtual de objetos que tem suas projeções nos eixos coordenados em intersecção (AGARWAL, 2008). O método *Sweep and Prune* (BARAFF, 1992) é baseado na varredura espacial e será apresentado a seguir.

3.2.1 *SWEEP AND PRUNE*

O método *Sweep and Prune* assume que cada objeto do ambiente virtual encontra-se envolvido por um volume, geralmente por caixas alinhadas aos eixos, e realiza o cálculo de intersecção das projeções destes volumes sobre os eixos coordenados (BARAFF, 1992).

Dois volumes envoltórios intersectam se e somente se suas projeções nos eixos X, Y e Z se sobrepõem. Pode-se dizer que o método realiza uma redução dimensional, testando

separadamente as projeções eixo por eixo (ERICSON, 2005). A Figura 3.6 mostra a projeção dos objetos de um ambiente nos eixos X e Y. Considera-se para este exemplo que o intervalo máximo e mínimo para cada eixo, correspondentes à dimensão do volume, são obtidos a partir da posição e da largura dos objetos.

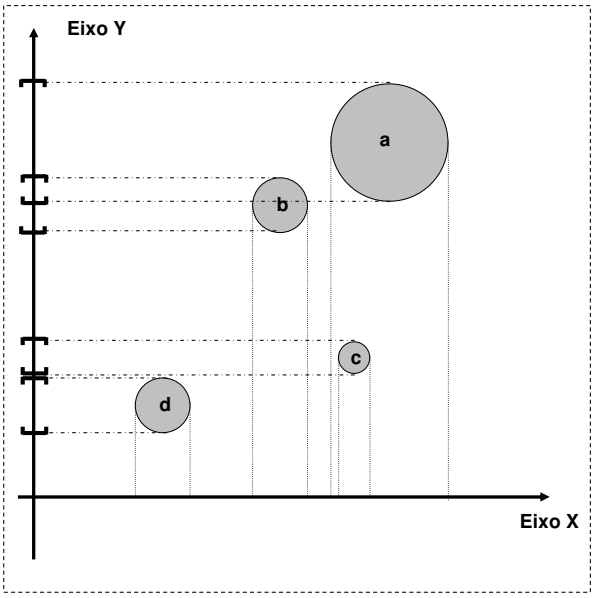


FIG. 3.6: Funcionamento do método *Sweep and Prune*: projeção de objetos nos eixos X e Y.

Como pode ser observado na Figura 3.6 os objetos a e c tem suas projeções e, sobreposição no eixo X. No entanto, não ocorre o mesmo no eixo Y. Com isso, podemos afirmar que os objetos não encontram-se em colisão.

Para identificar os pares de objetos que potencialmente estão colidindo, os pontos máximo e mínimo do volume envoltório de cada o objeto são mantidos em listas ordenadas, sendo uma lista para cada eixo. A ordenação das listas é realizada pelo algoritmo *insertion sort*.

De uma maneira geral, o *insertion sort* não é um algoritmo muito recomendado como um método de ordenação eficiente, pois pode exigir no pior caso $O(n^2)$ trocas. No entanto, ele acaba apresentando desempenho linear para a ordenação de uma lista que se encontra quase ordenada completamente. No método *Sweep and Prune*, após o movimento dos objetos no ambiente, as listas de cada eixo geradas necessitam ser atualizadas. Como os objetos se movem a pequenas distâncias de um determinado instante de tempo para outro,

a ordem dos elementos das listas não sofrem muitas variações. Com isso, a aplicação do *insertion sort* torna-se adequada para a ordenação das listas.

A partir das listas ordenadas deve-se então ser determinados, para cada eixo, todos os pares cujos intervalos se sobrepõem. Para isso, as listas de cada eixo devem ser varridas em busca destes pares.

Para a determinação dos objetos que tem as projeções dos sesu volumes envoltórios em sobreposição, o método *Sweep and Prune* mantém uma lista de intervalos ativos para cada eixo, inicialmente vazia. Estas listas vão sendo utilizadas à medida que a varredura vai sendo realizada nas listas ordenadas de cada eixo.

Se ao varrer uma lista ordenada de uma determinado eixo for encontrado um ponto mínimo de um objeto, este objeto é combinado com os objetos cujos pontos mínimos encontram-se armazenados na lista ativa, caso haja algum. Com isso, os pares são gerados e o ponto mínimo é incluído na lista de intervalos ativos. Se ao varrer a lista ordenada for encontrado um ponto máximo, o seu ponto mínimo correspondente é removido da lista ativa. Os pares finais resultam da intersecção entre os pares gerados para cada eixo.

A cada passo da simulação as listas associadas aos eixos têm que ser atualizadas devido aos objetos do ambiente se movimentarem. Neste caso, a coerência temporal do ambiente passa a ser explorada pelo *Sweep and Prune*.

Levando em consideração o fato de que objetos se movem a pequenas distâncias de um determinado instante de tempo para outro, pode-se supor que as listas de intervalos ordenadas estarão próximas de estarem ordenadas em um próximo instante. Dessa maneira as listas não necessitam ser completamente refeitas a cada passo de simulação e sim apenas atualizadas e se necessário ordenadas aplicando-se o algoritmo *Insertion Sort*.

Com isso, segundo Baraff (BARAFF, 1992), o custo de todo o processo seria $O(n+k)$, onde n é o número de objetos do ambiente e k é o número de objetos em intersecção.

3.3 CONSIDERAÇÕES

Apesar da otimização significativa que os métodos de particionamento espacial apresentam quando comparadas com o método *Força Bruta*, essas técnicas trazem também algumas limitações. No caso de *Grades Regulares*, a dificuldade consiste em estimar de maneira adequada para diferentes ambientes virtuais a melhor granularidade. Já para *Octrees*, *Kd-Trees* e *BSP-Trees* a dificuldade consiste em estimar a melhor profundidade da árvore usada e além disso conseguir árvores balanceadas.

Para uma mesma configuração de objetos, devido aos modos de divisão do espaço serem distintos, uma *Kd-Tree* apresenta um número menor de regiões que uma *Octree*. Além disso, *Kd-Trees* são extensíveis a espaços de maior número de dimensões. Por outro lado, a subdivisão regular das *Octrees* levam a diversos algoritmos eficientes para busca e construção.

Outro problema comum destes métodos é a quantidade de memória consumida para a execução dos mesmos. Como um objeto pode interceptar várias regiões simultaneamente, alternativas sugerem que o objeto seja duplicado para cada região ou sua geometria decomposta. Tais alternativas podem funcionar bem para objetos estáticos, no entanto para objetos dinâmicos elas se tornam computacionalmente caras devido às atualizações necessárias para cada movimento do objeto.

Além disso, muitas desses métodos assumem que, de certa forma, os objetos estão uniformemente dispersos no ambiente. Com isso, a idéia de reduzir a quantidade de pares de objetos a serem considerados nos testes de colisão está diretamente ligada a redução da região em que o objeto se encontra. Poderia-se assumir que reduzindo-se o tamanho da região a ser tratada, o número de objetos seria igualmente reduzido. Se, no entanto, muitos (ou todos) os objetos encontrarem-se aglomerados em uma mesma região do ambiente, o número de objetos a serem tratados seria muito grande, visto que os objetos estariam concentrados em uma única região.

O método *Sweep and Prune* tende a ter seu desempenho prejudicado em situações onde os objetos de um ambiente virtual encontram-se alinhados segundo um eixo específico, como mostra a Figura 3.7.

Neste tipo de configuração, os pontos mínimos e máximos dos volumes envoltórios de todos os objetos sobre o eixo em questão se encontram muito próximos uns dos outros. Isto faz com que mesmo que pequenos movimentos dos objetos sejam realizados de um instante para outro, a ordenação da lista de objetos do eixo pode deteriorar-se para $O(n^2)$ (ERICSON, 2005).

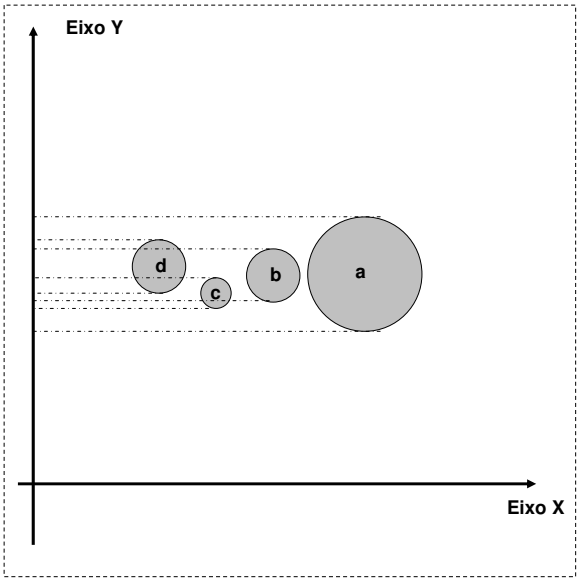


FIG. 3.7: Situação crítica ao desempenho do método *Sweep and Prune*: objetos alinhados no eixo Y.

4 MÉTODO DE DESCARTE BASEADO EM ÁREAS DE INTERESSE

Neste capítulo será definido o conceito de áreas de interesse e como é aplicado em Ambientes Virtuais Colaborativos (AVCs). Será também apresentado o método de descarte proposto, as definições adotadas e como o método funciona.

4.1 ÁREAS DE INTERESSE

O conceito de áreas de interesse tem sido muito utilizado para satisfazer requisitos de escalabilidade em Ambientes Virtuais Colaborativos (AVCs) (MORSE, 1996)(GREENHALGH, 1997)(OLIVEIRA, 2003). Esses ambientes apresentam usuários distribuídos geograficamente que podem interagir entre si e com o ambiente, podendo realizar alguma tarefa colaborativamente.

A necessidade do desenvolvimento de ambientes escaláveis vem do fato de que AVCs podem potencialmente ter um número elevado de usuários simultaneamente, e que isto pode facilmente sobrecarregar a rede como também impor um alto processamento.

Em um AVC mensagens são geradas a partir das interações dos usuários e trocadas entre eles para manter a consistência do ambiente. O fato de todos os usuários receberem mensagens de atualização uns dos outros no ambiente virtual traz uma situação que não garante uma boa escalabilidade. Desse modo, a redução do volume de mensagens trocadas torna-se essencial, a fim de evitar que a quantidade de usuários se torne um gargalo para o desempenho do ambiente como um todo.

Considerando que um usuário não está interessado nos eventos que ocorrem em todo o ambiente virtual, mas em apenas uma região específica ao redor de onde ele se encontra, Áreas de Interesse, podem ser definidas delimitando com quem um usuário poderá se comunicar. Cada usuário, então, passa a se comunicar apenas com as entidades que estão nas regiões que apresentem alguma interseção com a sua área de interesse, o que vem a reduzir consideravelmente a quantidade de mensagens trocadas.

4.1.1 GERENCIAMENTO DE ÁREAS DE INTERESSE

O gerenciamento de áreas de interesse em um ambiente virtual consiste em definir como serão representadas as regiões e onde objetos encontram-se localizadas, bem como a forma de armazenamento destes dados de maneira eficiente para que estes possam ser utilizados posteriormente. Neste contexto, o gerenciamento de áreas de interesse pode seguir duas abordagens: baseada em regiões ou baseada no objeto (VAN HOOK, 1994)(MORSE, 1996).

Na abordagem baseada em regiões o ambiente virtual é dividido em áreas (células) estáticas e a troca de mensagens somente é realizada entre usuários que se encontram dentro de uma célula ou em células vizinhas. Esta abordagem é adotada pelo NPSNET (MACEDONIA, 1994), onde o ambiente virtual é dividido em células hexagonais. A Figura 4.1 apresenta um ambiente particionado por regiões, onde pode-se notar que através da utilização de células hexagonais poucas células acabam tendo de ser tratadas em um determinado instante. Quando um objeto move-se no ambiente ele poderá apenas passar a ocupar uma das suas células vizinhas e com isso apenas um número limitado de células precisam ser atualizadas, como é o caso das células com bordas destacadas da Figura 4.1.

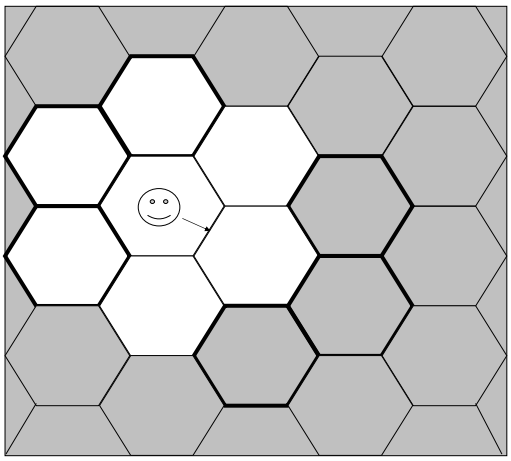


FIG. 4.1: Células hexagonais do NPSNET: ao mover-se, um objeto deve atualizar a sua vizinhança.

Na abordagem baseada em objetos tem-se um particionamento por objeto. Nesta abordagem cada objeto é associado a uma área, denominada Aura, que define uma região do ambiente virtual sobre a qual o objeto exercerá influência (BENFORD, 1993). Geral-

mente a aura é representada por uma esfera centrada no objeto, mas no entanto, qualquer forma geométrica pode ser adotada para representar uma área de interesse. Seja qual for a forma geométrica escolhida, a área de interesse sempre acompanha o objeto ao qual está associada caso ele se mova no ambiente. Como um objeto pode comunicar as suas ações somente para aqueles objetos que estiverem dentro da sua área de interesse, qualquer movimentação faz com que as áreas de interesse devam ser atualizadas de maneira a inserir ou remover objetos que estão ou não ao seu alcance. Esta abordagem é adotada pelo MASSIVE (GREENHALGH, 1997) e pelo VELVET (OLIVEIRA, 2003).

Na arquitetura VELVET (OLIVEIRA, 2003) áreas de interesse ajustáveis são utilizadas de modo a permitir que usuários de um ambiente possam reduzir ou estender a sua visão do ambiente de acordo com seu poder de processamento e características da rede, de modo a colaborarem eficientemente. Para que isso seja possível, VELVET trabalha com áreas de interesse esféricas que dinamicamente aumentam ou diminuem sua região conforme a densidade de objetos no ambiente, como mostra a Figura 4.2.

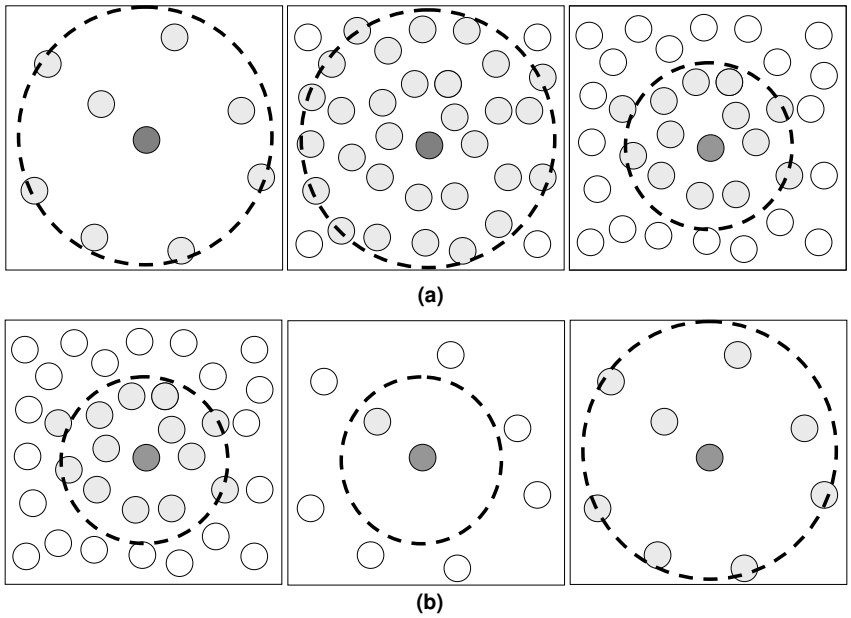


FIG. 4.2: Áreas de interesse ajustáveis do VELVET: em (a) o tamanho da área de interesse é reduzido e em (b) o tamanho da área de interesse é expandido devido a quantidade de usuários presentes na simulação.

Na Figura 4.2 áreas de interesse são representadas por esferas pontilhadas e os objetos representados por uma esfera preenchida em cinza escuro referem-se aos objetos que possuem uma área de interesse associada. Objetos representados por esferas preenchidas

em cinza claro estão em interseção com a área de interesse e desse modo são visíveis. Os objetos representados por esferas brancas não são de interesse.

Como pode ser visto na Figura 4.2 o tamanho das áreas de interesse é dinamicamente reduzido ou expandido de acordo com a quantidade de objetos a serem tratados. Uma alta densidade de objetos faz com que automaticamente a área de interesse seja reduzida de maneira que a quantidade de processamento necessária possa ser realizada eficientemente. O mesmo se aplica quando o número de objetos diminui, onde neste caso a área de interesse é expandida de modo que mais objetos possam se tornar visíveis.

A abordagem para gerenciamento de áreas de interesse baseada em objetos reduz consideravelmente o volume de mensagens a serem trocadas em um ambiente virtual se comparada a abordamem baseada em regiões (VAN HOOK, 1994). Isso se deve ao fato de que a abordagem baseada em áreas de interesse possui um particionamento do ambiente vinculado à localização do objeto e não a um ambiente como um todo, como o que ocorre com a abordagem baseada em regiões.

4.2 O MÉTODO DE DESCARTE *AOIP*

4.2.1 DEFINIÇÕES

O método de descarte proposto, denominado *AoIP* (*Area of Interest Partitioning*), faz uso do conceito de áreas de interesse seguindo a abordagem baseada em objetos apresentada em (OLIVEIRA, 2003). Para o desenvolvimento do método foram consideradas as seguintes definições:

- O ambiente virtual pode ser formado por objetos tridimensionais rígidos dinâmicos ou estáticos.
- e é a quantidade de objetos estáticos presentes no ambiente virtual.
- d é a quantidade de objetos estáticos presentes no ambiente virtual.
- OE é o conjunto formado por todos os objetos estáticos presentes no ambiente virtual, denotado por $OE = \{OE_0,\ldots,OE_{e-1}\}$. Um objeto estático é caracterizado por possuir um tamanho, um volume envoltório e uma posição.
- OD é o conjunto formado por todos os objetos dinâmicos presentes no ambiente virtual, denotado por $OD = \{OD_0,\ldots,OD_{d-1}\}$. Um objeto dinâmico é caracterizado

por possuir um tamanho, um volume envoltório, uma posição, uma velocidade e uma direção.

- Os objetos que compõem o conjunto OD possuem velocidades iguais e constantes.

- VE é o conjunto formado pelos volumes envoltórios associados aos objetos estáticos do conjunto OE, denotado por $VE = \{VE_0,\ldots,VE_{e-1}\}$. A cada objeto OE_i do conjunto OE, onde $0 \leq i < e$, está associado um volume envoltório VE_i, tal que VE_0 está associado a OE_0, VE_1 está associado a $OE_1,\ldots$, VE_{e-1} está associado a OE_{e-1}.

- VD é o conjunto formado pelos volumes envoltórios associados aos objetos dinâmicos do conjunto OD, denotado por $VD = \{VD_0,\ldots,VD_{d-1}\}$. A cada objeto OD_i do conjunto OD, onde $0 \leq i < d$, está associado um volume envoltório VD_i, tal que VD_0 está associado a OD_0, VD_1 está associado a $OD_1,\ldots$, VD_{d-1} está associado a OD_{d-1}.

- AI é o conjunto formado pelas áreas de interesse, denotado por $AI = \{AI_0,\ldots,AI_{d-1}\}$. A cada objeto dinâmico OD_i do conjunto OD, onde $0 \leq i < d$, está associada uma área de interesse AI_i, tal que AI_0 está associado a OD_0, AI_1 está associado a $OD_1,\ldots$, AI_{d-1} está associado a OD_{d-1}.

- Uma área de interesse possui uma forma geométrica e é caracterizada por um tamanho fixo, uma posição, uma velocidade e uma direção de acordo com o objeto a que está associada. Tanto o tamanho quanto a forma geométrica definida para uma área de interesse influenciam diretamente a quantidade de pares gerados para a fase de descarte. Quanto menor é o tamanho de uma área de interesse e mais ajustável a área for ao formato do objeto, maior a redução de pares.

- Qualquer objeto do ambiente virtual pode, em um determinado instante, estar dentro de uma área de interesse. A cada área de interesse AI_i do conjunto AI, onde $0 \leq i < d$, está associado um conjunto T, denotado por $T^i = \{T^0,\ldots,T^{k-1}\}$, formado por $k \geq 0$ objetos pertencentes aos conjuntos OE e OD que estão parcial ou totalmente contidos na área de interesse AI_i associada ao objeto OD_i.

- Um objeto encontra-se parcialmente ou totalmente em uma área de interesse se existe intersecção entre o seu volume envoltório e a área de interesse correspondente.

A operação de interseção entre uma área de interesse do conjunto AI e um volume envoltório do conjunto VE ou VD consiste na interseção entre a forma geométrica adotada para a área de interesse e a forma geométrica adotada para o volume envoltório.

- P é o conjunto dos pares de objetos resultante do método *AoIP*, denotado por $P = \{P_0,\ldots,P_{m-1}\}$, formado por $m \geq 0$ pares. Os pares pertencentes ao conjunto P são formados pela combinação do objeto OD_i associado a uma área de interesse AI_i com cada objeto pertencente ao conjunto T^i correspondente.

4.2.2 FUNCIONAMENTO

O método *AoIP* tem como dados de entrada os objetos (estáticos e/ou dinâmicos) inseridos no ambiente virtual. O primeiro passo do método consiste em criar as áreas de interesse para cada objeto (estático e/ou dinâmico) do ambiente. Cada área de interesse é criada a partir da especificação da forma geométrica a ser considerada e os parâmetros específicos a este tipo de área. Por exemplo, para uma área de interesse esférica é necessário a especificação de seu raio, bem como o centro do objeto a que esta área será associada.

De uma maneira geral, o método *AoIP* realiza testes de interseção entre as áreas de interesse e os volumes envoltórios dos objetos do ambiente virtual. Estes testes tem a finalidade de verificar quais objetos estão dentro de uma determinada área de interesse de modo a inserir elementos no conjunto T^i, inicialmente vazio. A partir dos elementos do conjunto T^i, são obtidos os pares de objetos a serem repassados para a fase de refinamento.

Cada vez que um objeto movimenta-se é necessária a atualização do conjunto T^i criado para a área de interesse associada a este objeto. Essa atualização consiste em adicionar os novos objetos que se encontram na área de interesse e remover aqueles que não estão mais ao seu alcance.

Definidos os conjuntos T^i para cada área de interesse o próximo passo é obter os pares de objetos resultantes que serão processados pela fase de refinamento. A obtenção do conjunto P é realizada através da combinação entre os objetos de cada conjunto T^i e o objeto ao qual a área de interesse a que pertence este conjunto está associada. O pseudocódigo 1 apresenta o método *AoIP*.

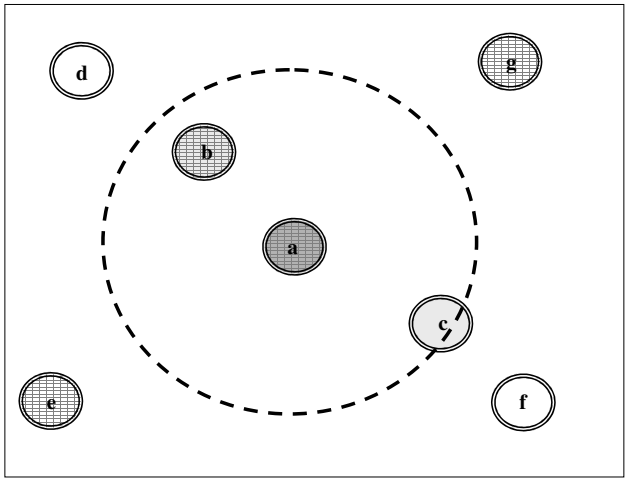
Pseudocódigo 1 Método *AoIP*

```
AoIP(OE, OD)
        para i=0 até d-1 faça
                T^i = ∅
                para j=0 até e-1 faça
                        se(VerificarInterseção(AI_i,VE_j)) então
                                T^i = T^i ∪ {OE_j}

                para k=0 até d-1 faça
                        se(VerificarInterseção(AI_i,VD_k)) então
                                T^i = T^i ∪ {OD_k}

        para i=0 até d-1 faça
                para j=0 até k-1 faça
                        P = P ∪ {(OD_i,T^i_j)}
```

A Figura 4.3 apresenta os pares resultantes de uma área de interesse para uma ambiente com objetos estáticos e dinâmicos.



OD = {a, b, e, g} **OE = {c, d, f}**
AI_1 = {a}
T_1 = {b, c}
P = {(a,b), (a,c)}

FIG. 4.3: Geração de pares para uma área de interesse esférica, assumindo que apenas 'a' possui área de interesse.

Neste exemplo apenas dois objetos intersectam a área de interesse, os objetos b e c. Desse modo, somente estes objetos serão considerados nos testes de colisão realizados posteriormente na fase de refinamento, como pode ser visto pelos pares resultantes inseridos no conjunto P.

5 AMBIENTE DE SIMULAÇÃO

Um ambiente de simulação foi criado de maneira a simular o comportamento na Fase de Descarte de alguns dos métodos de particionamento espacial e varredura espacial apresentados no Capítulo 3, como também o comportamento do método desenvolvido utilizando o conceito de áreas de interesse. Neste capítulo será apresentado como este ambiente de simulação foi definido.

5.1 IMPLEMENTAÇÃO DO AMBIENTE DE SIMULAÇÃO

O ambiente de simulação foi desenvolvido utilizando o Motor EnCIMA (MALFATTI, 2008; SANTOS, 2008) implementado em C++ e OpenGL. A execução do ambiente de simulação é baseada em um laço principal que é executado várias vezes no decorrer do tempo. Cada iteração do laço principal foi chamado de passo de simulação.

Para a obtenção de dados refentes ao comportamento dos métodos testados optou-se por utlizar um ambiente de simulação composto por apenas objetos dinâmicos. Para facilitar a obtenção destes dados a execução do simulador foi estruturada em três etapas que são processadas a cada passo de simulação:

a) Obtenção das novas posições dos objetos dinâmicos inseridos no ambiente: a cada passo de simulação os objetos dinâmicos do ambiente virtual movimentam-se, e novas posições são geradas para estes objetos.

b) Realização da fase de descarte: esta etapa consiste em utilizar métodos de descarte para gerar o conjunto de pares de objetos que serão passados para a Fase de Refinamento onde os testes de colisão serão realizados. Para a fase de descarte cada método foi estruturado da seguinte maneira: criação das estruturas de dados, atualização e geração dos pares de colisão. Os métodos de descarte utilizados pelo simulador e suas características de implementação são apresentadas na Seção 5.1.2 seguindo esta estruturação.

c) Realização da fase de refinamento: nesta etapa as colisões entre objetos são identificadas.

O ambiente de simulação consiste de uma caixa desenhada em *wireframe* (COHEN, 2006) com altura, largura e profundidade de 4 metros. Neste ambiente foram inseridas esferas vermelhas representando os objetos dinâmicos. Cada esfera é definida por 256 triângulos, cujo raio possui tamanho variando entre 1% a 10% do valor do tamanho das dimensões do ambiente.

Os objetos dinâmicos possuem velocidade linear constante igual a 0.01 metros/quadro e movem-se de acordo com um vetor direção $\vec{d}$ cujas componentes são valores randômicos no intervalo $[-1; 1]$, gerados pelo ambiente de simulação. Quando um objeto colide com uma das paredes do ambiente, a componente do vetor direção $\vec{d}$ que corresponde ao eixo da parede a qual o objeto colidiu é invertida. Dessa forma, o objeto segue com um movimento simétrico em um sentido oposto.

Cada objeto do ambiente é envolvido por uma volume esférico (*BoundingSphere*), apresentado na Seção 2.1.1.1. Optou-se por um volume esférido devido a este tipo de volume se ajustar melhor à forma do objeto escolhido.

Neste ambiente, a colisão entre os objetos é representada da seguinte maneira: quando é identificada uma colisão entre duas ou mais esferas, a cor original vermelha das mesmas é alterada para a cor verde enquanto estiver colidindo, permitindo que as esferas se interpenetrem. A Figura 5.1 mostra o ambiente de simulação composto por 10 objetos envolvidos por *BoundingSpheres*.

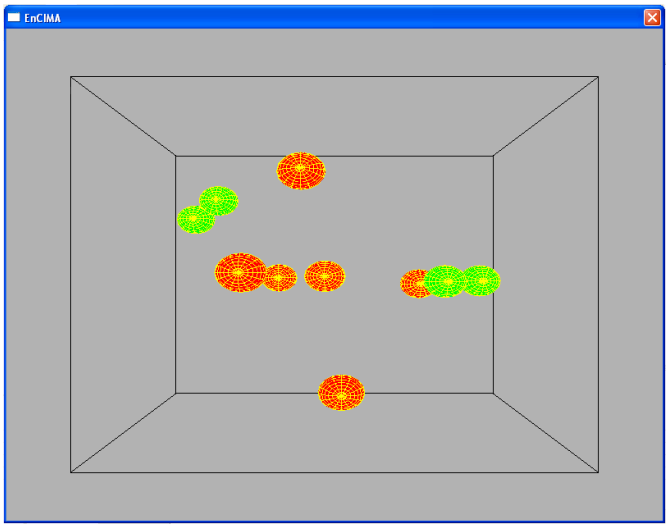


FIG. 5.1: Ambiente de Simulação.

Os objetos do ambiente virtual são armazenados em um vetor dinâmico (*Vector*) que será utilizado por todos os métodos de descarte implementados.

5.1.1 ARQUIVO DE CONFIGURAÇÃO

De modo a facilitar a realização dos experimentos foi definido um arquivo de configuração para o ambiente de simulação. Neste arquivo podem ser configurados os seguintes itens:

a) Número de passos de simulação: configura a quantidade de vezes que o laço principal será executado.

b) Quantidade de objetos dinâmicos.

c) Método de descarte a ser utilizado: podem ser escolhido entre *Força Bruta*, *Grade Regular*, *BSP-Tree*, *Sweep and Prune* e *AoIP*.

De acordo com o método de descarte escolhido, as seguintes configurações podem ser realizadas:

- *Grade Regular*: configurar a quantidade de células para cada dimensão da caixa.
- *BSP-Tree*: configurar a profundidade da árvore binária.
- *Sweep and Prune*: configurar a quantidade de eixos a serem considerados na geração de pares de objetos.
- *AoIP*: configurar a calibração da área de interesse.

5.1.2 IMPLEMENTAÇÃO DOS MÉTODOS DE DESCARTE

Dos métodos apresentados no Capitulo 3, além do método *Força Bruta*, foi escolhido um método de cada categoria para ser implementado. Para simular a fase de descarte com particionamento espacial não hiérarquico, o método *Grade Regular* foi implementada. O método *BSP-Tree* foi implementado como representante do particionamento espacial hiérarquico. E para simular a fase de Descarte com varredura espacial o método *Sweep and Prune* foi implementado. Essas implementações basearam-se em (ERICSON, 2005).

A seguir serão descritos detalhes de implementação de cada método selecionado, bem como do método AoIP apresentado no Capítulo 4.

5.1.2.1 FORÇA BRUTA

Para este método a fase de descarte foi implementada da seguinte maneira:

- Criação: nenhuma estrutura de dados é criada.
- Atualização: nenhuma atualização de estrutura de dados é realizada.
- Geração de Pares: combinação dos objetos do ambiente dois a dois, não sendo consideradas combinações entre um mesmo objeto e nem pares repetidos.

5.1.2.2 GRADES REGULARES

Para este método a fase de descarte foi implementada da seguinte maneira:

- Criação: é criada a grade regular e são geradas as suas células, as quais são representadas por caixas. Para cada dimensão do ambiente virtual é considerada uma quantidade de células obtida do arquivo de configuração. O tamanho das dimensões de cada célula é obtido dividindo-se o tamanho de cada dimensão do ambiente pela quantidade de células definida para cada respectiva dimensão. A partir do tamanho das dimensões da célula e do tamanho das dimensões do ambiente são obtidos os pontos máximo e mínimo de cada célula de modo a posicioná-la no ambiente. As células são armazenadas em um vetor dinâmico (*Vector*). Criadas as células, é determinada a relação de pertinência entre os objetos e células. Para cada célula os objetos que encontram-se em sua região são armazenados em um vetor dinâmico (*Vector*). Objetos que pertencem a mais de uma célula devem ser duplicados.
- Atualização: a grade regular e suas células são criadas uma única vez durante a inicializaçao da aplicação. A cada passo de simulação somente são atualizados os vetores de objetos que encontram-se em cada célula.
- Geração de Pares: os pares de objetos são obtidos através da combinação dos objetos de cada célula dois a dois.

5.1.2.3 BSP-TREE

Para este método a Fase de Descarte foi implementada da seguinte maneira:

- Criação: é criada a árvore binária a partir dos objetos do ambiente e a profundidade obtida do arquivo de configuração. Para a criação da árvore binária é selecionado um plano de divisão com posição e orientação arbitrários. A posição do plano é calculada com base no centróide dos objetos e a orientação é obtida a partir da normal dos objetos. Após calculado o plano de divisão os objetos do ambiente são classificados quanto a este plano. Objetos que encontram-se na frente do plano são adicionados ao nó esquerdo da árvore. Objetos que encontram-se atrás do planos são adicionados ao nó direito da árvore. Aqueles objetos que encontram-se em ambos os lados são duplicados para o nó da esquerda e o nó da direita. A árvore é construída recursivamente até que a profundidade especificada seja alcançada.

- Atualização: a árvore binária é destruída e construída novamente a cada passo de simulação.

- Geração de Pares: os pares de objetos são obtidos através da combinação dos objetos de cada nó-folha dois a dois.

5.1.2.4 SWEEP AND PRUNE

Para este método a fase de descarte foi implementada da seguinte maneira:

- Criação: é criado um vetor dinâmico (*Vector*) para cada eixo com os intervalos de pontos mínimos e máximos de todos os objetos do ambiente. Esse vetor é, então, ordenado utilizando o algoritmo *Insertion Sort*.

- Atualização: o método *sweep and prune* explora a coerência temporal e desse modo os vetores que armazenam os intervalos de cada objeto não necessitaram ser reconstruídos a cada passo de simulação. Os intervalos são somente atualizados e é aplicado o algoritmo *Insertion Sort* para possíveis ordenações que necessitem ser realizadas.

- Geração de Pares: os eixos a serem considerados para a geração de pares são obtidos no arquivo de configuração. Especificado o valor 1 o eixo X é considerado, especificado o valor 2 os eixos X e Y são considerados, e especificado o valor 3 os eixos X, Y e Z são considerados. Na geração de pares é realizada uma varredura em busca de objetos que têm seus intervalos em sobreposição. Cada par de objetos encontrado

é armazenado em um vetor dinâmico de pares que é mantido para cada eixo. Após obter os vetores de pares para cada eixo é montado o vetor de pares finais: para o eixo X este já é considerado o conjunto final de pares, para os eixos X e Y os pares finais são aqueles pares que estão contidos em ambos os vetores e para os 3 eixos os pares finais são aqueles que encontram-se repetidos nos vetores dos 3 eixos.

5.1.2.5 AOIP

A implementação do método AoIP seguiu as definições apresentadas na Seção 5.1.2.5 do Capitulo 4. A forma geométrica adotada para a área de interesse foi uma esfera. Para este método a Fase de Descarte foi implementada da seguinte maneira:

- Criação: são criadas as áreas de interesse para cada objeto dinâmico do ambiente. Para cada área de interesse são realizados testes de interseção com os volumes envoltórios dos objetos do ambiente, no caso testes de interseção entre esferas. Duas esferas estão intersectando se as distâncias entre os seus centros é menor do que a soma de seus raios (ERICSON, 2005), como ilustra a Figura 5.2. Objetos que que encontram-se visíveis a uma área de interesse são armazenados em um vetor dinâmico (*Vector*).

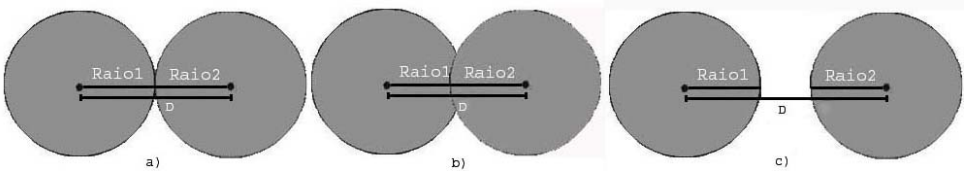


FIG. 5.2: Intersecção entre esferas: em (a) e (b) as esferas estão em intersecção ($D \leq Raio1 + Raio2$); em (c) as esferas não estão em intersecção ($D > Raio1 + Raio2$).

- Atualização: para cada objeto dinâmico é criada uma área de interesse durante a inicialização da aplicação. A cada passo de simulação somente são atualizados os vetores de objetos que encontram-se em cada área de interesse.

- Geração de Pares: os pares de objetos são obtidos através da combinação dos objetos que encontram-se em cada área de interesse com o objeto a que está associada esta área, sendo descartados os pares repetidos.

A Figura 5.3 mostra o ambiente de simulação composto por 10 objetos e suas áreas de interesse com raio de duas vezes o tamanho do raio do objeto.

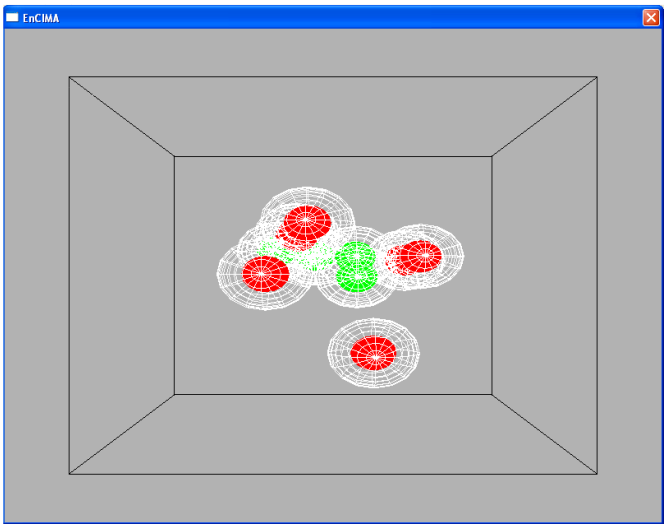


FIG. 5.3: Visualização das áreas de interesse no ambiente de simulação.

5.1.3 IMPLEMENTAÇÃO DA FASE DE REFINAMENTO

Nos Métodos de Descarte implementados os volumes envoltórios dos objetos são utilizados para verificar se um objeto encontra-se dentro de uma determinada região do ambiente. Na Fase de refinamento os volumes envoltórios são utilizados nos testes de interseção para verificar se um objeto encontra-se em colisão com outro.

Apesar da Fase de Refinamento ter como objetivo realizar testes de interseção mais exatos a nível de polígonos, optou-se por utilizar os volumes envoltórios para os testes por questões de simplificação.

6 EXPERIMENTOS E RESULTADOS

Para os experimentos realizados durante o desenvolvimento deste trabalho foi utilizado o ambiente de simulação descrito no Capítulo 5. Neste capítulo serão apresentadas a metodologia seguida para tais experimentos e a análise dos resultados obtidos.

6.1 METODOLOGIA

Para a realização dos experimentos foram definidas as variáveis dependentes e independentes a serem consideradas. A Figura 6.1 apresenta uma visão geral da forma com que os experimentos foram realizados.

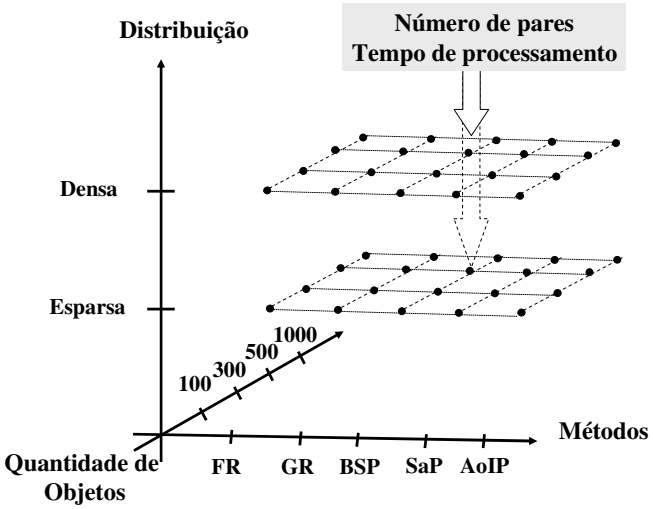


FIG. 6.1: Variáveis dependentes e independentes analisadas no ambiente de simulação: os círculos pretos representam as variáveis dependentes e os eixos coordenados representam as variáveis independentes.

As variáveis dependentes referem-se ao número de pares gerados e ao tempo de processamento obtido para cada método de descarte implementado descrito na Secção 5.1.2 do Capítulo 5. Sendo um dos objetivos desta Dissertação comparar os números de pares gerados e os tempos de processamento obtidos, diferentes simulações foram realizadas, onde foram consideradas as seguintes variáveis independentes:

- A quantidade de objetos em movimento: 100, 300, 500 e 1000 objetos.

- A distribuição dos objetos no ambiente: densa e esparsa.
- O método de descarte utilizado: *Força Bruta*, *Grades Regulares*, *BSP-Tree*, *Sweep and Prune* e *AoIP*. Para cada método de descarte foram realizadas as seguintes configurações de acordo com suas particularidades:
  - Para o método *Grades Regulares* foram testadas configurações diferentes para a quantidade de células considerada para cada dimensão do ambiente: 2x2x2, 4x4x4, 1x2x1 e 2x2x1, como mostra a Figura 6.2 .
  - Para o método *BSP-Tree* foram testadas profundidades diferentes para a árvore: 2, 3, 5 e 6.
  - Para o método *Sweep and Prune* foram testadas configurações diferentes para a quantidade de eixos a ser considerada na varredura: eixo X, eixos X e Y e eixos X, Y e Z.
  - Para o método *AoIP* foram testadas configurações diferentes para a calibraçao da área de interesse: raio da área de mesmo tamanho do raio do objeto, raio da área 2 vezes o tamanho do raio do objeto e raio da área 3 vezes o tamanho do raio do objeto. É importante salientar que definido o tamanho para uma área de interesse, este é aplicado a todos os objetos do ambiente.

Os testes foram realizados em um computador com processador Intel Core 2 Duo, 2GB de memória RAM e placa de vídeo NVidia GForce 7900 GT/GTO. Os números de pares gerados e os tempos de processamento foram obtidos baseados na média de várias execuções realizadas para cada método. Cada execução consistiu de 500 passos de simulação, conforme descrito na seção 5.1, onde os dados obtidos foram registrados em arquivos de *log* e analisados graficamente.

O tempos de processamento analisados consistiram nos tempos obtidos pela fase de descarte, incluindo as etapas de criação, atualização e geração de pares. O tempos de processamento para a fase de refinamento não foram medidos, considerando-se que a quantidade de pares gerados pela fase de descarte tem influência direta na quantidade de tempo de processamento da fase de refinamento. Desse modo, assume-se que à medida que mais pares são gerados pela fase de descarte de um método, mais será custoso o processamento da fase de refinamento.

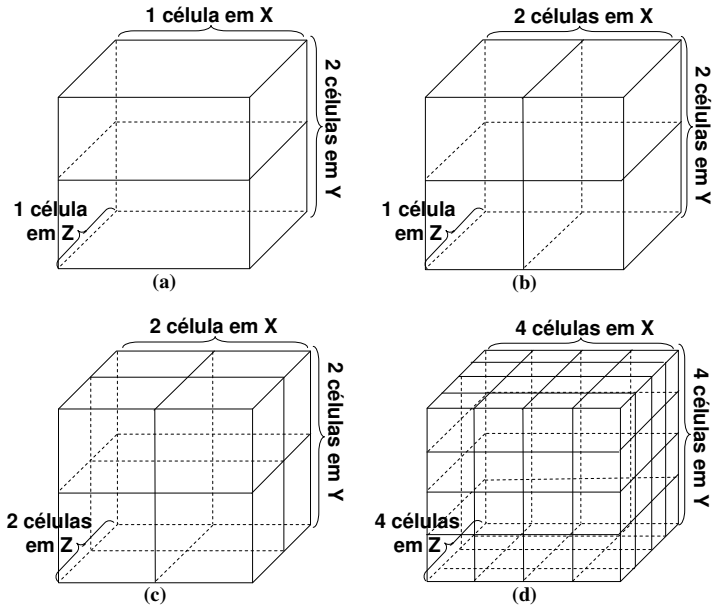


FIG. 6.2: Configurações utilizadas para o método *Grades Regulares*: em (a) grade com 2 células (GR1x2x1), em (b) grade com 4 células (GR2x2x1), em (c) grade com 8 células (GR2x2x2) e em (d) grade com 16 células (GR4x4x4).

Os resultados gerados pelo método *Força Bruta* foram utilizados como ponto de referência na análise dos resultados obtidos pelos métodos de descarte testados, devido serem considerados críticos para o processo de detecção de colisão.

Devido a similaridade do padrão de comportamento dos resultados obtidos para a variação da quantidade de objetos inseridos no ambiente, neste Capítulo irá ser apresentada a análise dos resultados considerando uma simulação com 100 objetos. Os resultados obtidos para as demais variações de objetos são apresentados no Apêndice 9.2.

6.2 DESCRIÇÃO DOS CENÁRIOS

Os experimentos realizados consistiram em simular dois tipos de cenários: um cenário com distribuição espacial densa de objetos e um cenário com distribuição espacial esparsa de objetos.

A caracterização de um cenário em denso ou esparso foi determinada pelo cálculo da densidade dos objetos em cada passo de simulação. Em uma distribuição espacial densa os objetos encontram-se num nível maior de aglomeração, estando mais próximos uns dos outros. Já para uma distribuição espacial esparsa, os objetos encontram-se mais dispersos uns dos outros.

Para o cenário com distribuição espacial densa, os objetos iniciam todos na mesma posição (origem do ambiente) e podem movimentar-se em uma espaço delimitado por uma caixa de 2x2x2 dentro do ambiente de dimensões 4x4x4. O cenário com distribuição espacial esparsa é obtido a partir do cenário denso onde, neste caso, os objetos encontram-se aglomerados e vão dispersando-se no ambiente, podendo movimentar-se livremente. A Figura 6.3 mostra a simulação dos dois cenários simulados para 100 objetos:

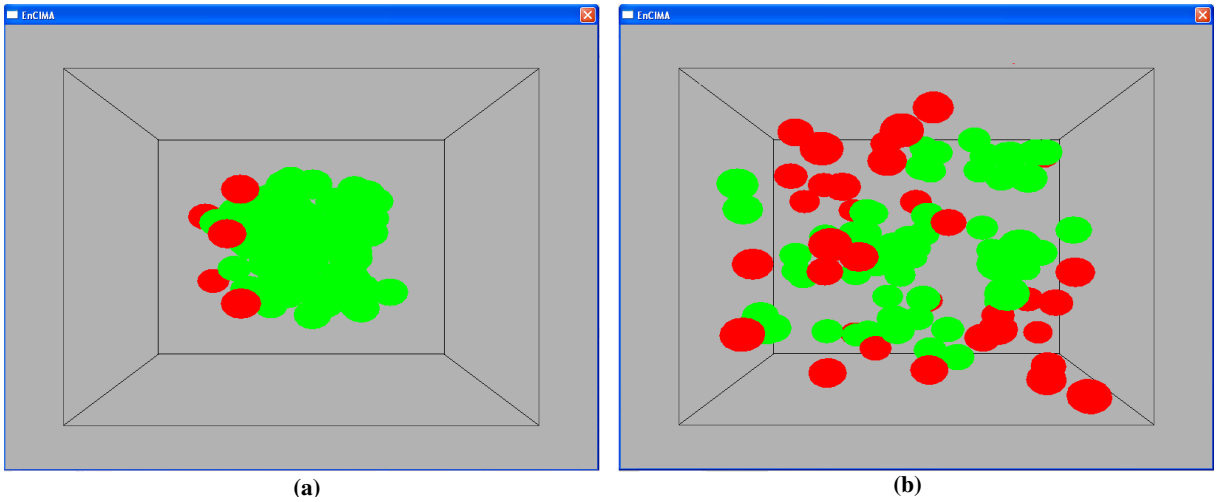


FIG. 6.3: Distribuição espacial para os cenários simulados: em (a) uma distribuição densa dos objetos e em (b) uma distribuição esparsa dos objetos.

Para cada passo de simulação a densidade dos objetos foi obtida calculando-se a distância média de todos os objetos ao seu centróide, obtido a partir de suas posições e os pesos relacionados ao seu tamanho. Quanto maior o raio de um objeto maior a sua influência sobre o posicionamento do centróide.

Quanto menor o valor da densidade encontrado, mais próximos os objetos encontram-se no ambiente. Os gráficos da Figura 6.4 apresentam, para os cenários simulados, as densidades obtidas para 100 objetos em 500 passos de simulação. Os resultados obtidos para as densidades das demais simulações com 300, 500 e 1000 objetos podem ser encontradas no Apêndice 9.1.

Nas próximas seções serão analisados os resultados obtidos quanto ao número de pares gerados e o tempo de processamento obtido nos dois tipos de cenários simulados.

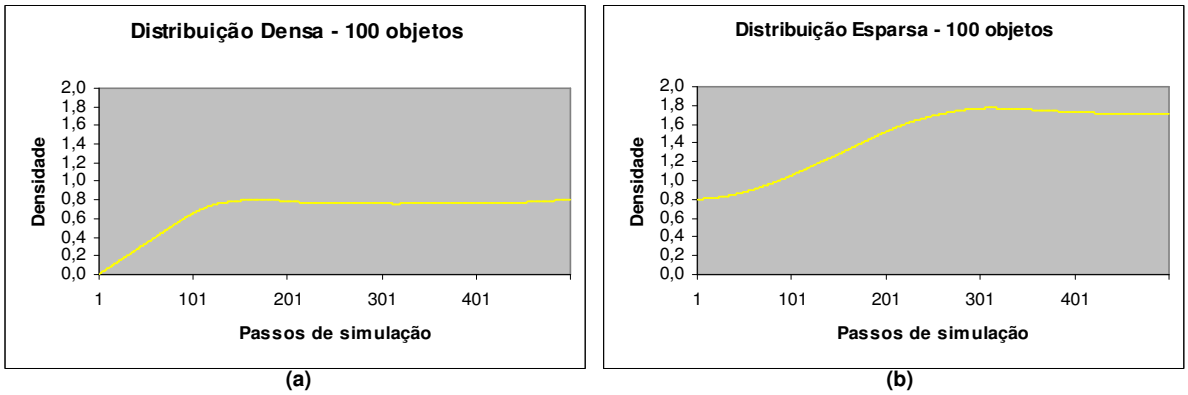


FIG. 6.4: Densidades dos cenários simulados para 100 objetos: em (a) distribuição densa dos objetos e em (b) distribuição esparsa dos objetos.

6.3 NÚMERO DE PARES GERADOS NA DISTRIBUIÇÃO DENSA

Nesta seção serão apresentados os resultados obtidos quanto à geração de pares pelos métodos de descarte para a distribuição densa dos objetos simulada.

6.3.1 MÉTODO *FORÇA BRUTA*

O gráfico da Figura 6.5 apresenta a variação das médias de pares gerados para o método *Força Bruta* no ambiente de simulação com 100, 300, 500 e 1000 objetos. A Tabela 6.1 apresenta os valores das médias obtidas.

TAB. 6.1: Média de pares gerados pelo método *Força Bruta* - distribuição densa

	100	**300**	**500**	**1000**
FB	**4.950**	**44.850**	**124.750**	**499.500**

No gráfico da Figura 6.6 pode ser observado mais detalhadamente os pares gerados a partir de 100 objetos para cada distribuição em cada passo de simulação.

Para este método a movimentação dos objetos no ambiente não influencia a quantidade de pares gerados, mantendo-se esta quantidade constante durante toda a simulação. Sendo n o número de objetos inseridos no ambiente, a quantidade de pares gerados para o método *Força Bruta* é calculada por $(n(n-1))/2$.

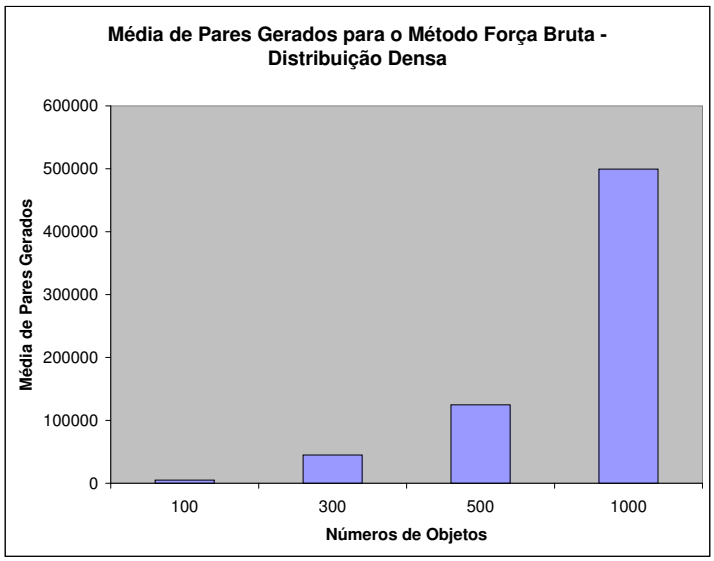


FIG. 6.5: Média de pares gerados pelo método *Força Bruta* para uma distribuição densa dos objetos.

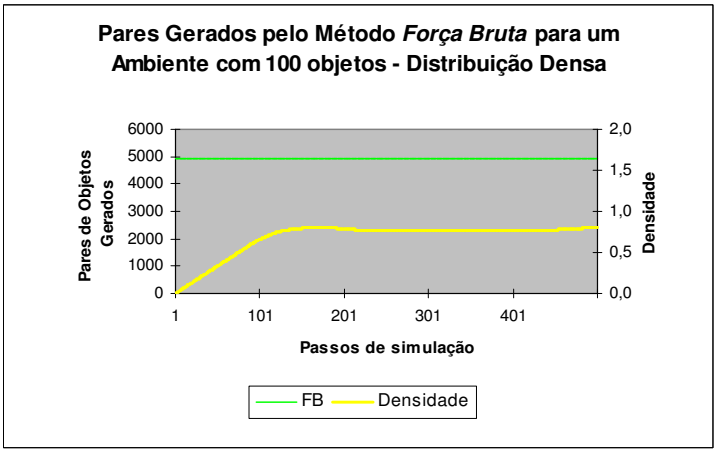


FIG. 6.6: Número de pares gerados pelo método *Força Bruta* para uma distribuição densa com 100 objetos.

6.3.2 MÉTODO *GRADES REGULARES*

O gráfico da Figura 6.7 apresenta a variação das médias de pares gerados para as configurações adotadas para o método *Grades Regulares* no que se refere a quantidade de células criadas para o ambiente de simulação com 100, 300, 500 e 1000 Objetos. A Tabela 6.2 apresenta os valores das médias obtidas para cada configuração.

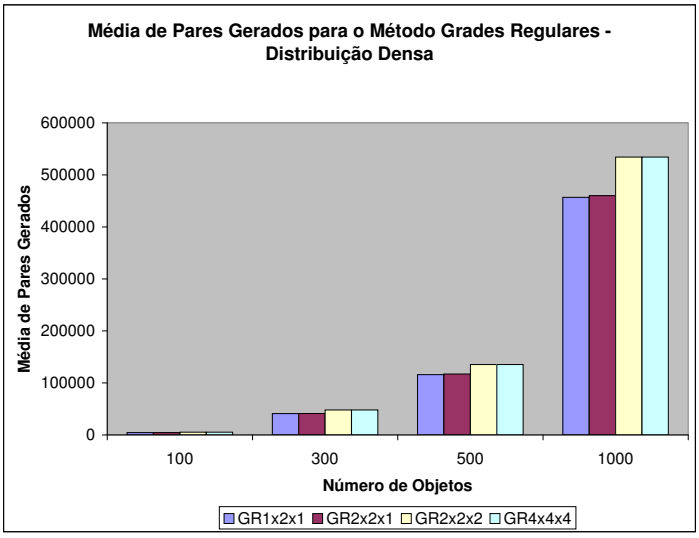


FIG. 6.7: Média de pares gerados pelo método *Grades Regulares* para uma distribuição densa dos objetos.

TAB. 6.2: Média de pares gerados pelo método *Grades Regulares* - distribuição densa

	100	**300**	**500**	**1000**
GR1x2x1	**4.399,4**	**41.080,6**	**115.885,5**	**456.788,6**
GR2x2x1	4.522,4	41.252,1	117.082,6	460.001,9
GR2x2x2	5.183,9	48.042,9	135.265,6	534.208,9
GR4x4x4	5.183,9	48.042,9	135.265,6	534.208,9

Para este cenário a configuração que mostrou-se mais eficiente quanto a redução de pares gerados foi a grade definida com 1 célula para os eixos X e Z, e 2 células para o eixo Y (GR1x2x1). No entanto, como pode ser observado na Tabela 6.2, não houve uma diferença muito significativa entre os valores das médias de pares obtidas para cada configuração de acordo com as variações da quantidade de objetos inseridos no ambiente.

Nos gráficos da Figura 6.8 pode ser observado mais detalhadamente os pares gerados a partir de 100 objetos em cada passo de simulação.

Supõe-se que quanto maior for a quantidade de células em uma grade, menor será o número de pares gerados. No entanto, segundo testes realizados para a distribuição densa

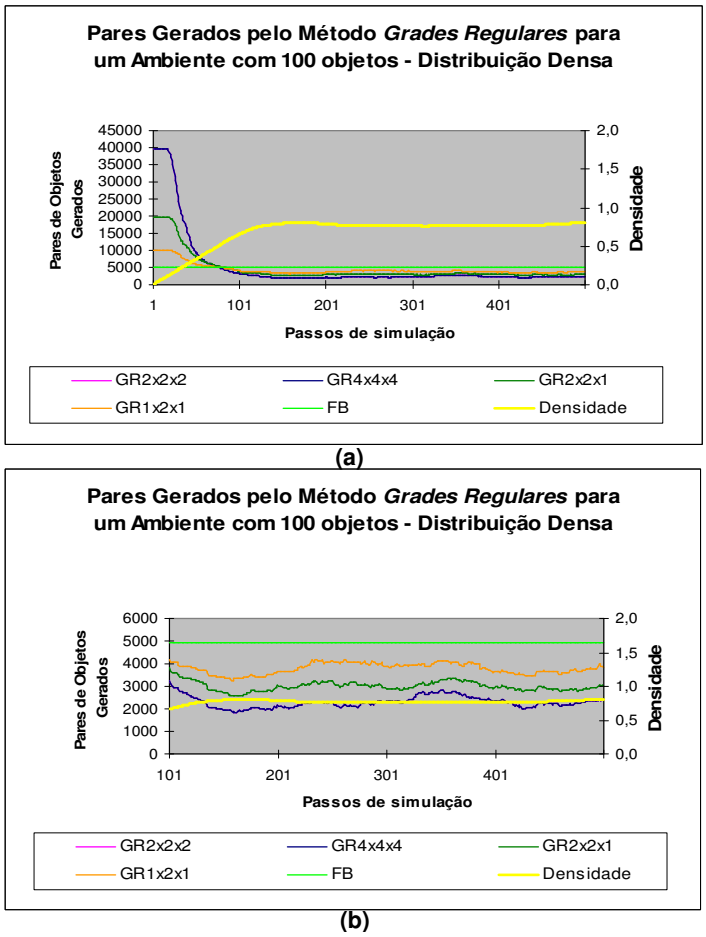


FIG. 6.8: Número de pares gerados pelo método *Grades Regulares* para uma distribuição densa com 100 objetos: em (a) a simulação completa com 500 passos e em (b) a simulação com os 100 passos iniciais descartados.

de objetos essa suposição não se connfirmou. De acordo com as configurações adotadas, mostradas na Figura 6.2, a configuração GR4x4x4 deveria gerar o menor número de pares, mas produz o mesmo efeito da configuração GR2x2x2, o qual também não é o mais eficiente.

Podemos observar que as configurações GR2x2x2 e GR4x4x4 geram a mesma quantidade de pares devido a todos os objetos encontrarem-se concentrados nas proximidades do centro do ambiente. Com isso, constata-se que haviam poucos objetos nas 2 células das extremidades de cada dimensão da configuração 4x4x4, resultando, deste modo, no mesmo efeito produzido pela configuração GR2x2x2.

As configurações GR2x2x2 e GR2x2x1 não foram eficientes na redução de pares devido a duplicação de objetos realizada. Com isso, a configuração GR1x2x1, apesar de possuir o menor número de células, gerou o menor número de pares por criar poucos objetos duplicados.

É interessante notar na Figura 6.8 que os objetos vão se dispersando ao longo de cada passo de simulação e a duplicação de objetos vai sendo reduzida. No primeiros passos de simulação quando o ambiente ainda encontra-se bastante denso, a quantidade de pares obtida acaba sendo superior à quantidade obtida pela *Força Bruta*. À medida que os objetos vão se dispersando a quantidade de pares gerados cai para valores inferiores ao da *Força Bruta*.

Considerando as configurações adotadas para o método *Grades Regulares* e a densidade dos objetos para cada passo de simulação pode-se concluir que quanto mais aglomerados os objetos encontram-se no ambiente e quanto mais células uma grade conter, maior será a duplicação realizada em uma distribuição densa de objetos.

6.3.3 MÉTODO *BSP-TREE*

A Figura 6.9 mostra a variação das médias de pares gerados para as configurações do método *BSP-Tree* no que se refere a profundidade da árvore criada para o ambiente de simulação com 100, 300, 500 e 1000 Objetos. A Tabela 6.3 apresenta os valores das médias obtidas para cada configuração.

Para este cenário a configuração que mostrou-se mais eficiente quanto a redução de pares gerados foi a profundidade 2 (BSP2). No gráfico da Figura 6.10 pode ser observado mais detalhadamente os pares gerados a partir de 100 objetos em cada passo de simulação.

Quanto aos resultados obtidos dos testes realizados com o método *BSP-Tree* pode-se

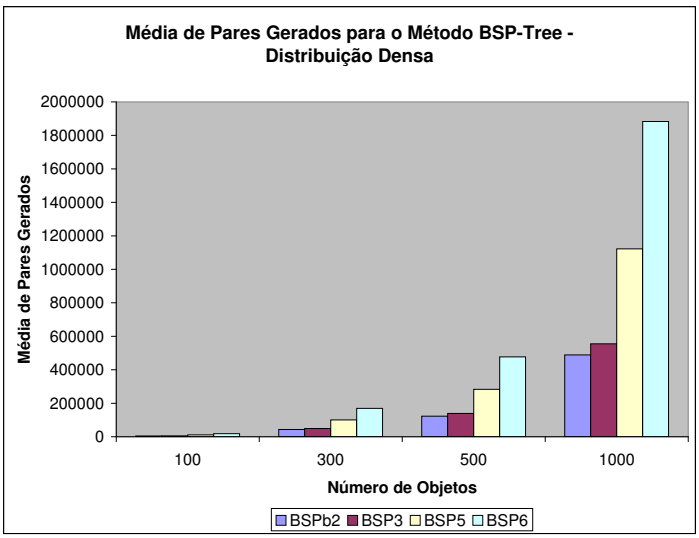


FIG. 6.9: Média de pares gerados pelo método *BSP-Tree* para uma distribuição densa dos objetos.

TAB. 6.3: Média de pares gerados pelo método *BSP-Tree* - distribuição densa

	100	**300**	**500**	**1000**
BSP2	**4.760,3**	**43.684,1**	**122.936,1**	**488.911,2**
BSP3	5.390,2	49.388,2	139.162,8	555.415,1
BSP5	10.960,4	100.649,9	282.900,2	1.122.297,2
BSP6	18.481,2	169.659,2	476.774,7	1.883.039,1

concluir que há muita duplicação de objetos se comparadas as médias da quantidade de pares gerados com as médias obtidas pela *Força Bruta*. Esta duplicação é diretamente influenciada pela profundade escolhida para a árvore. Quanto maior a profundidade, em mais regiões o ambiente será dividido, e consequentemente, menores serão as regiões obtidas. Desse modo, em uma distribuição densa regiões muito pequenas irão resultar em um alto índice de duplicação devido a probabilidade maior de existir objetos posicionadas nas fronteiras das regiões. Por esse motivo, a configuração que eficientemente reduziu os pares de objetos foi a com profundidade 2, seguida pela profundidade 3, 5 e 6.

Atenção especial deve ser dada ao comportamento apresentado pelas configurações BSP5 e BSP6. Nestes casos, além da quantidade de pares ser superior à quantidade gerada pela *Força Bruta* nos primeiros passos de simulação, nos passos posteriores há variações bruscas dos seus comportamentos. No caso da configuração BSP6 há instantes em que a quantidade de pares gerados também se mantém superior à quantidade gerada pela *Força Bruta*. Essas variações podem ser explicadas pela maior duplicação que há

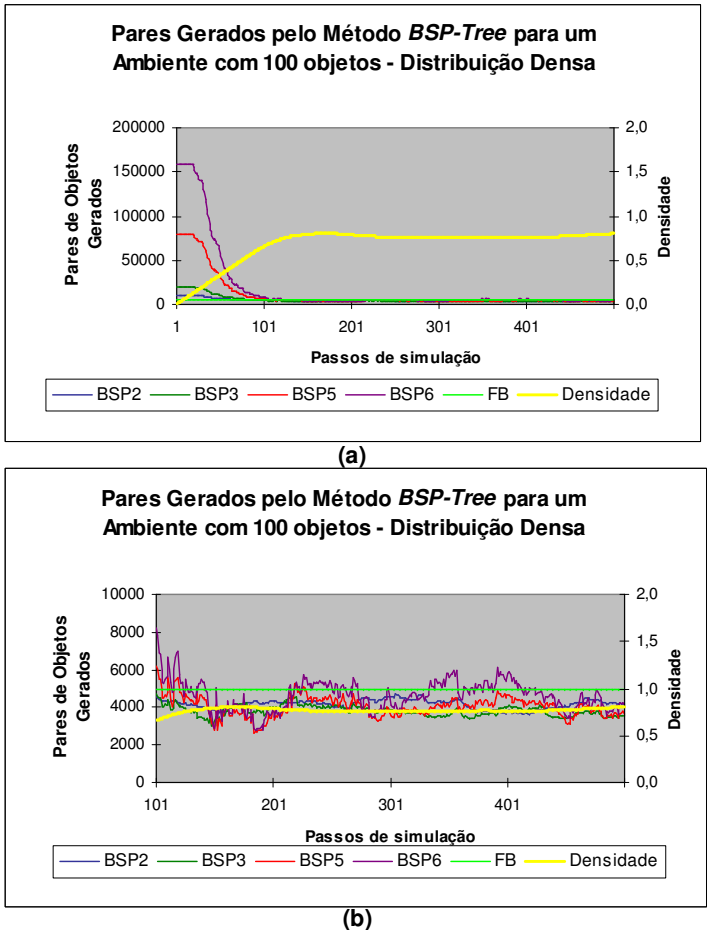


FIG. 6.10: Número de pares gerados pelo método *BSP-Tree* para uma distribuição densa com 100 objetos: em (a) a simulação completa com 500 passos e em (b) a simulação com os 100 passos iniciais descartados.

devido às regiões serem menores.

O fato de a seleção do plano de divisão ser de orientação e posicionamento arbitrários faz com que regiões não tenham um tamanho uniforme, como ocorre com as células de uma *Grade Regular*. Os cálculos para o posicionamento e orientação do plano de divisão consideram a localização dos objetos no ambiente de forma a buscar uma árvore balanceada, e com isso não há garantias que duplicações não sejam obtidas.

6.3.4 MÉTODO *SWEEP AND PRUNE*

O gráfico da Figura 6.11 apresenta a variação das médias de pares gerados para as configurações do método *Sweep and Prune* no que se refere a quantidade de eixos escolhidos para varredura do ambiente de simulação com 100, 300, 500 e 1000 Objetos. A Tabela 6.4 apresenta os valores das médias obtidas para cada configuração.

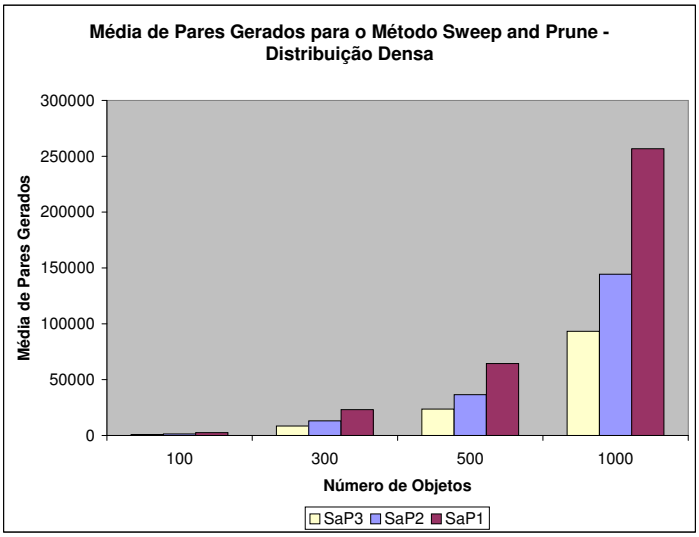


FIG. 6.11: Média de pares gerados pelo método *Sweep and Prune* para uma distribuição densa dos objetos.

TAB. 6.4: Média de pares gerados pelo método *Sweep and Prune* - distribuição densa

	100	**300**	**500**	**1000**
SaP3	**913,2**	**8.471,9**	**23.589,3**	**93.291,1**
SaP2	1.407,3	13.078,5	36.497,8	144.422,3
SaP1	2.565,6	23.165,9	64.448,3	256.763,8

Para este cenário a configuração que mostrou-se mais eficiente quanto a redução de pares gerados foi a que levou em consideração os 3 eixos coordenados (SaP3). No gráfico

da Figura 6.12 pode ser observado mais detalhadamente os pares gerados a partir de 100 objetos em cada passo de simulação.

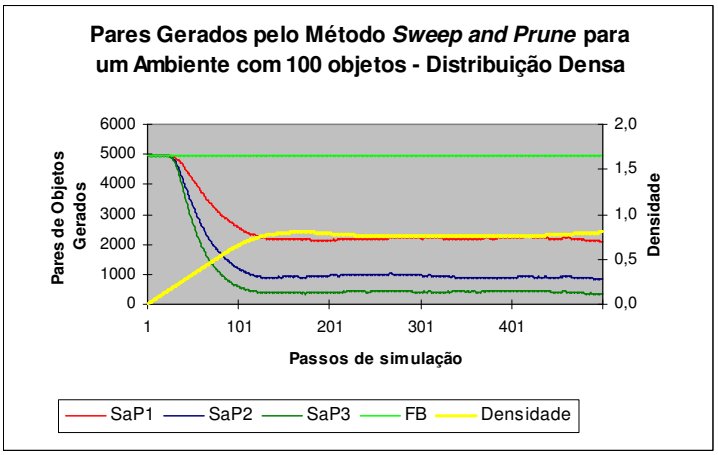


FIG. 6.12: Número de pares gerados pelo método *Sweep and Prune* para uma distribuição densa com 100 objetos.

Para o método *Sweep and Prune* a configuração mais eficiente quanto a quantidade de pares gerados, independente do cenário a que for aplicado, é considerar os três eixos para varredura. Essa configuração gera somente os pares de objetos que tem seus volumes envoltórios em colisão. No entanto, isso não quer dizer que os objetos estejam em colisão. A possibilidade da colisão estar ocorrendo vai depender de quão bem o volume utilizado se ajusta ao objeto. Considerar somente 1 ou 2 eixos podem resultar em uma quantidade igual ou maior de pares gerados.

Como era de se esperar, o gráfico da Figura 6.12 mostra que para uma distribuição densa de objetos utilizando 3 eixos para varredura resulta uma quantidade muito menor de pares de objetos gerados em comparação às outras configurações. É interessante também observar que os resultados apresentados para os primeiros passos de simulação, onde os objetos encontram-se em um alto nível de aglomeração, os resultados obtidos igualam-se aos obtidos pela *Força Bruta*. À medida que os objetos vão se afastando uns dos outros os resultados tornam-se melhores que os obtidos pela *Força Bruta*.

6.3.5 MÉTODO *AOIP*

O gráfico da Figura 6.13 mostra a variação das médias de pares gerados para as configurações do método *AoIP* no que se refere ao tamanho da área considerada para o ambiente de simulação com 100, 300, 500 e 1000 Objetos. A Tabela 6.5 apresenta os

valores das médias obtidas para cada configuração.

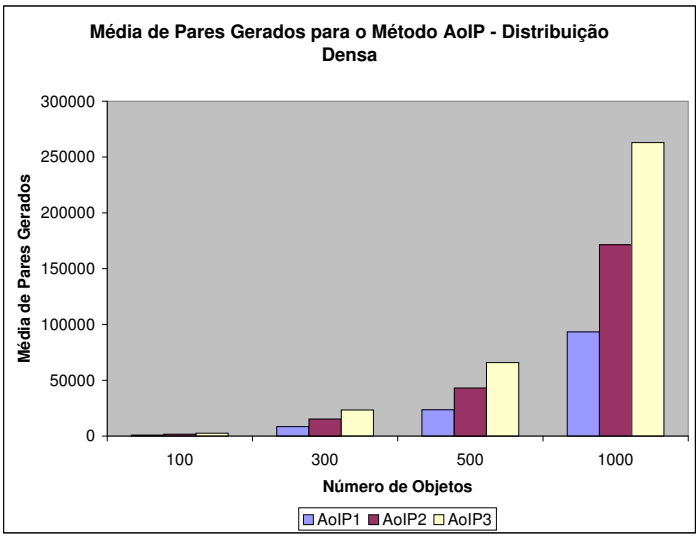


FIG. 6.13: Média de pares gerados pelo método *AoIP* para uma distribuição densa dos objetos.

TAB. 6.5: Média de pares gerados pelo método *AoIP* - distribuição densa

	100	**300**	**500**	**1000**
AoIP1	**913,2**	**8.471,8**	**23.589,3**	**93.291,1**
AoIP2	1.663,8	15.317,9	43.019,2	171.403
AoIP3	2.557,8	23.367,3	65.815,1	262.980,9

Para este cenário a configuração que mostrou-se mais eficiente quanto a redução de pares gerados foi a que considera uma área de interesse de tamanho igual ao tamanho do objeto (AoIP1). No gráfico da Figura 6.14 podem ser observados mais detalhadamente os números de pares gerados pelo algoritmo a partir de 100 objetos em cada passo de simulação.

Para o método *AoIP* quanto mais a área de interesse definida se ajusta ao objeto, menor é a quantidade de pares gerados. Da mesma forma que para o *Sweep and Prune*, essa configuração também gera somente os pares de objetos em que os seus volumes já encontram-se em colisão.

Como pode ser observado no gráfico da Figura 6.14, para uma distribuição densa de objetos, aumentando o tamanho da área de interesse, as quantidade de pares gerados também aumenta. Desse modo, utilizando para área de interesse um tamanho igual ao tamanho do objeto resulta em uma quantidade muito menor de pares de objetos gerados em comparação às outras configurações.

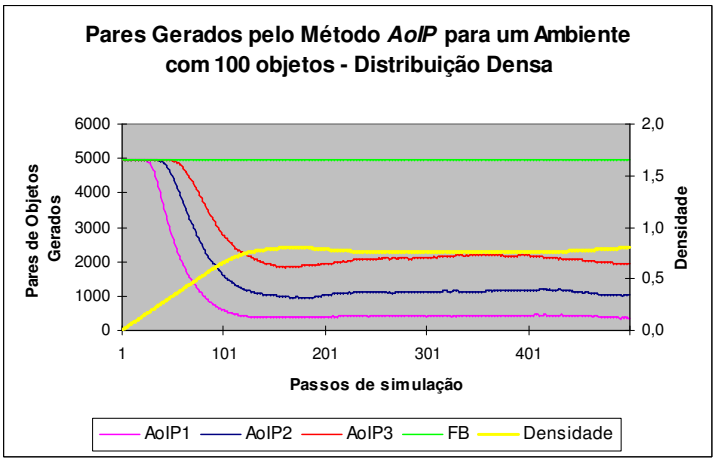


FIG. 6.14: Número de pares gerados pelo método *AoIP* para uma distribuição densa com 100 objetos.

Neste método, quanto mais aglomerados os objetos estiverem, maior é a probabilidade dos resultados se igualarem à *Força Bruta*, como é mostrado no gráfico da Figura 6.14 para os primeiros passos de simulação. À medida que os objetos se dispersam os resultados tornam-se melhores que os obtidos pela *Força Bruta*.

6.3.6 CONSIDERAÇÕES SOBRE OS RESULTADOS

O gráfico da Figura 6.15 mostra a variação das médias de pares gerados para os métodos testados e suas configurações.

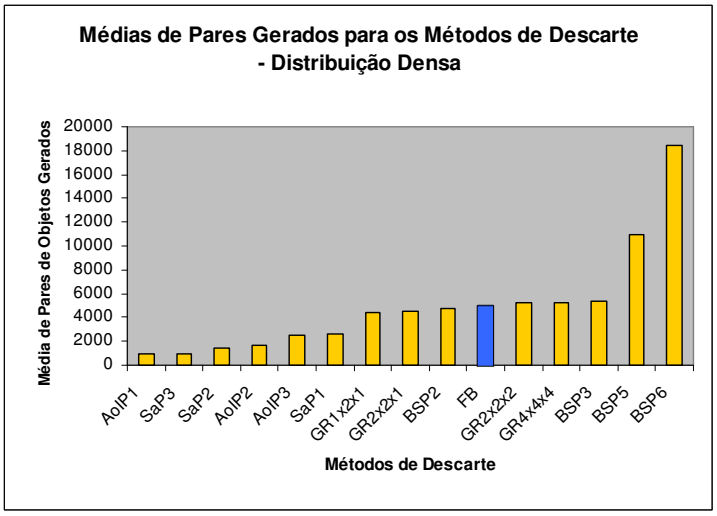


FIG. 6.15: Média de pares gerados pelos métodos de descarte para uma distribuição densa com 100 objetos.

Pode-se verificar que alguns dos métodos testados não são eficientes quando considerada uma distribuição densa dos objetos. Algumas configurações utilizadas para os métodos *BSP-Tree* e *Grades Regulares* resultaram em um número de pares gerados maior do que o gerado pela *Força Bruta*, devido às duplicações de objetos realizadas. Para o método *BSP-Tree* as configurações para profundidade de 5 e 6 obtiveram o maior número de duplicações entre todos os métodos. Para o método *Grades Regulares* o fato dos objetos concentrarem-se em poucas células fez com que estas tratassem uma grande quantidade de objetos.

Os métodos *AoIP* e *Sweep and Prune* obtiveram a maior redução do número de pares de objetos a serem considerados nos testes de colisão, considerando todas as configurações adotadas. O gráfico da Figura 6.16 apresenta o comportamento desses e dos demais métodos que descartaram a maior quantidade de pares de objetos em comparação à *Força Bruta*.

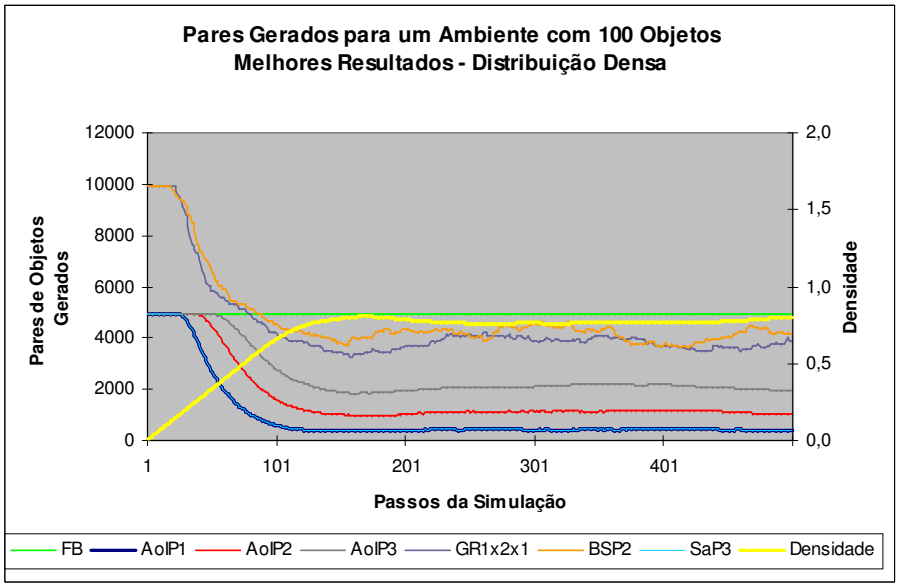


FIG. 6.16: Melhores resultados obtidos para uma distribuição densa com 100 objetos.

Para os métodos *AoIP* e *Sweep and Prune*, independente da configuração adotada, quanto mais aglomerados encontraram-se os objetos do ambiente, mais próximos da *Força Bruta* foram os resultados obtidos, como pode ser visto no gráfico da Figura 6.17.

Outra questão interessante quanto a estes dois métodos é que geraram a mesma quantidade de pares quando considerada uma área de interesse de mesmo tamanho do objeto e considerado os 3 eixos para o *Sweep and Prune*.

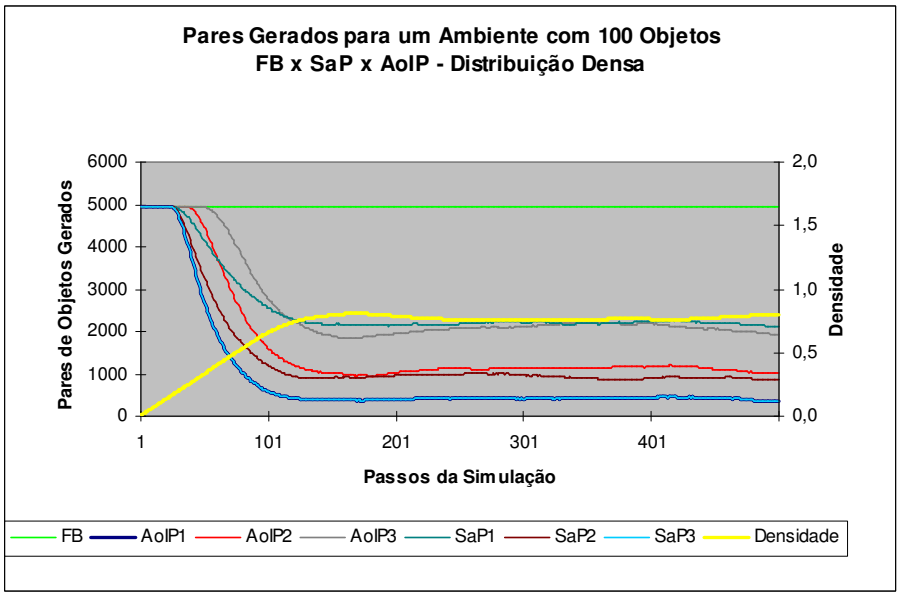


FIG. 6.17: Comparação entre os resultados obtidos para os métodos *AoIP* e *Sweep and Prune* para uma distribuição densa com 100 objetos.

6.4 TEMPOS DE PROCESSAMENTO OBTIDOS NA DISTRIBUIÇÃO DENSA

Nesta seção serão apresentados os tempos médios de processamentos obtidos na Fase de Descarte para cada método simulado em uma distribuição densa com 100, 300, 500 e 1000 objetos.

Como definido na Seção 5.1 do Capítulo 5 os métodos implementados para a fase de descarte foram divididos em três etapas: criação, atualização e geração de pares. O comportamento apresentado por cada método durante as simulações está relacionado com a maneira com que foram implementadas essas etapas. Os resultados gráficos, discutidos a seguir para cada método, são apresentados nos Apêndice 9.3 e seguem esta estruturação.

6.4.1 MÉTODO *FORÇA BRUTA*

Para a *Força Bruta*, independente da quantidade de objetos inseridos no ambiente, todo o tempo de processamento concentra-se na geração de pares de objetos, visto que

para este método não há estruturas a serem criadas e nem atualizadas. Portanto, para este método o tempo de processamento é proporcional ao número de objetos.

6.4.2 MÉTODO *GRADES REGULARES*

Grades Regulares possuem um tempo de criação relativamente baixo, pelo fato de que o particionamento do ambiente é realizado a partir de operações matemáticas simples. Esse método consome mais tempo nas etapas de atualização e geração de pares do que na etapa de criação, independente da quantidade de objetos inseridos no ambiente.

De uma maneira geral, observou-se que quanto maior o número de células na grade, maior foi o tempo de atualização das estruturas. Isso se deve à reconstrução, a cada passo de simulação, das listas de objetos que encontram-se em cada célula.

Quanto à etapa de geração de pares, pode-se observar que as configurações com um maior número de células tiveram um maior tempo devido às duplicações realizadas e a grande quantidade de objetos processados.

6.4.3 MÉTODO *BSP-TREE*

Para o método *BSP-Tree* os tempos de criação também foram relativamente baixos se considerados em comparação aos tempos de atualização e geração de pares.

De uma maneira geral, a etapa de atualização foi a que consumiu a maior quantidade de processamento devido a árvore da BSP-Tree ser reconstruída a cada passo de simulação. Notou-se que quanto maior foi a profundidade definida, maior foi o tempo total para a etapa de atualização.

Para a etapa de geração pode-se perceber que à medida que a quantidade de objetos aumentou e maior foi a profundidade definida, maior foi o tempo de processamento obtido. Isso ocorreu devido ao percurso até os nós-folhas da árvore ser maior. Considerando que quanto maior a profundidade definida, menores são as regiões em que o ambiente é dividido, pode-se dizer que para um ambiente denso, a duplicação de objetos será maior, e consequentemente, maior será a quantidade de objetos a ser processada por cada região.

6.4.4 MÉTODO *SWEEP AND PRUNE*

Para o *Sweep and Prune* é consumido mais tempo de processamento na etapa de geração de pares, pois é nesta etapa que ocorre a varredura do ambiente e o descarte

de objetos que não tem suas projeções em sobreposição. Os tempos obtidos para as etapas de criação e atualização foram relativamente baixos quando comparados ao tempo da etapa de geração de pares. A principal razão pela qual este método consome pouco processamento na etapa de atualização é o fato que ele não refaz as listas de objetos para os eixos considerados. No entanto, para ambientes densos há uma maior probabilidade de haverem objetos alinhados em um derteminado eixo, o que pode elevar o tempo de processamento para a etapa de atualização.

Como era de se esperar, os maiores tempos para a Fase de Descarte utilizando o método *Sweep and Prune* foram obtidos pela configuração que utiliza 3 eixos para varredura, seguida pela que considera 2 eixos e a que considera 1 eixo.

6.4.5 MÉTODO *AOIP*

Para o método *AoIP* a etapa de criação também não obteve tempos de processamento muito expressivos. O maior tempo de processamento ocorre na etapa de atualização, e principalmente, na etapa de geração de pares. Observou-se que à medida que a quantidade de objetos aumentou, houve a necessidade de um maior processamento para a geração de pares.

Como era esperado, à medida que aumentou-se o tamanho da área de interesse o tempo de processamento também aumentou, devido à quantidade extra de objetos a serem processados.

6.4.6 CONSIDERAÇÕES SOBRE OS RESULTADOS

Os gráficos da Figura 6.18 apresentam a relação entre os tempos médios de processamento e a média de pares de objetos gerados pelos métodos de descarte a partir das configurações específicas adotadas para cada método.

De acordo com os resultados apresentados não há relação entre a quantidade de pares gerados e a quantidade de tempo de processamento necessária para a fase de descarte. Um número menor de objetos gerados não significa que a Fase de Descarte teve um menor tempo de processamento. A quantidade de pares gerados irá ter influência direta na quantidade de tempo de processamento da fase de refinamento. Quanto mais pares forem gerados pela fase de descarte, mais será custoso o processamento da fase de refinamento.

Como pode ser observado nos gráficos da Figura 6.18, apesar dos métodos *AoIP* e *Sweep and Prune* gerarem os menores números de pares de objetos, estes métodos foram

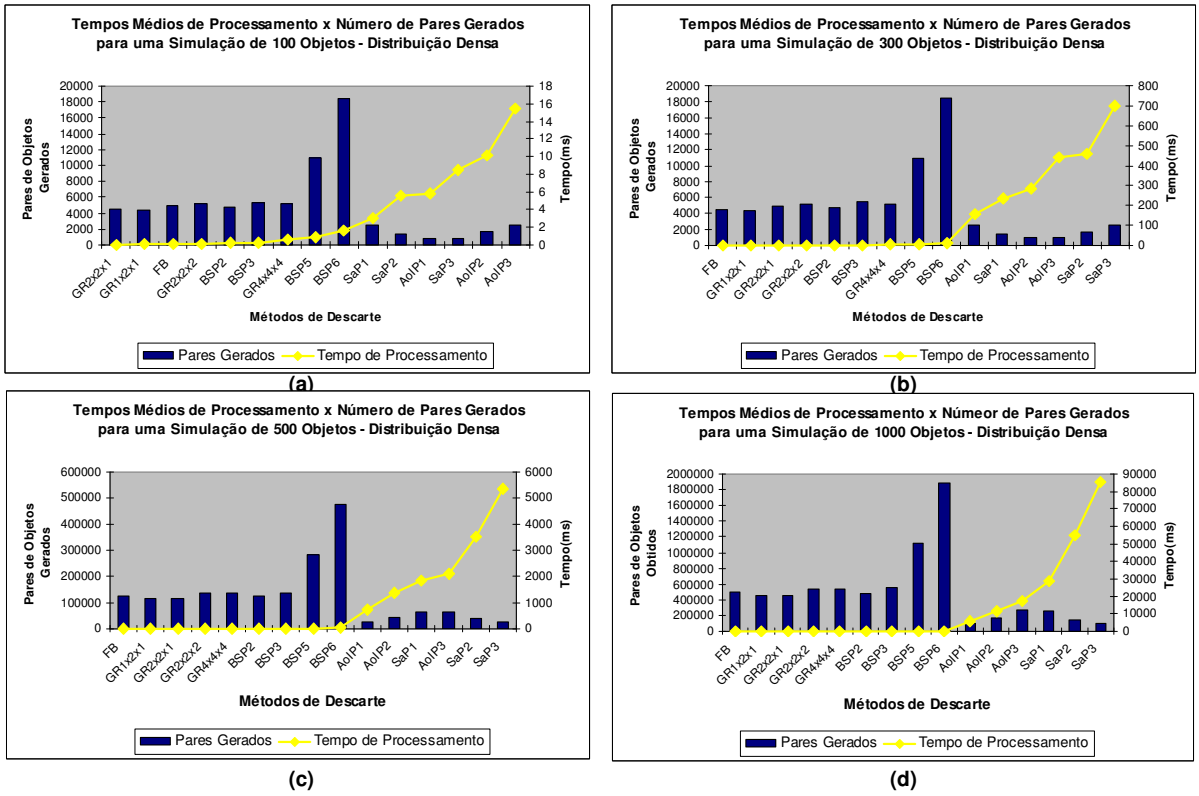


FIG. 6.18: Tempos médios de processamento e média do número de pares gerados para uma distribuição densa: (a) valores obtidos para 100 objetos, (b) valores obtidos para 300 objetos, (c) valores obtidos para 500 objetos e (d) valores obtidos para 1000 objetos.

os que apresentaram os maiores tempos médios de processamento para a Fase de Descarte. O método *Sweep and Prune* com varredura em 3 eixos (SaP3) foi o que apresentou o maior tempo de processamento quanto maior foi a quantidade de objetos considerada. *Grades Regulares* e a *Força Bruta* encontram-se entre os métodos que apresentaram os menores tempos para a fase de descarte.

Como foi analisado na Seção 6.3 os métodos que tiveram a maior redução dos pares de objetos para um ambiente denso, independente da quantidade de objetos inseridos, foram o Método *AoIP* com tamanho igual ao do objeto e *Sweep and Prune* com 3 eixos de varredura. Para os tempos de processamento obtidos para esses métodos em uma distribuição densa, é interessante observar que à medida que a quantidade de objetos aumenta, o tempo de processamento da Fase de Descarte para o Método *Sweep and Prune* aumenta consideravelmente se comparado ao Método *AoIP*.

6.5 NÚMERO DE PARES GERADOS NA DISTRIBUIÇÃO ESPARSA

Nesta seção serão apresentados os resultados obtidos quanto à geração de pares pelos métodos de descarte para a distribuição esparsa dos objetos simulada.

6.5.1 MÉTODO *FORÇA BRUTA*

A *Força Bruta* é um método que apresenta valores constantes para a quantidade de pares gerados, independente da quantidade de objetos e da movimentação desses objetos em cada passo de simulação. Isso ocorre para qualquer quantidade de objetos considerada. Desse modo, para a simulação com distribuição esparsa são considerados os mesmos resultados apresentados para a simulação com distribuição densa quanto ao número de pares gerados.

6.5.2 MÉTODO *GRADES REGULARES*

O gráfico da Figura 6.19 mostra a variação das médias de pares gerados para as configurações adotadas para o método *Grades Regulares* no que se refere a quantidade de células criadas para o ambiente de simulação com 100, 300, 500 e 1000 Objetos. A Tabela 6.6 apresenta os valores das médias obtidas para cada configuração.

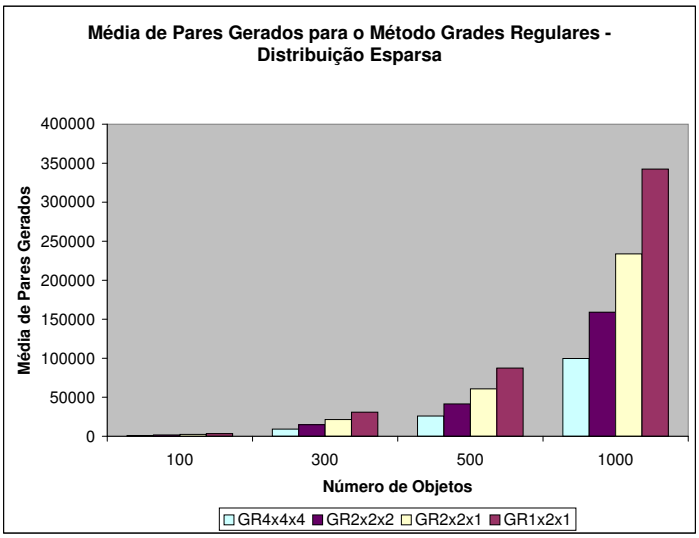


FIG. 6.19: Média de pares gerados pelo método *Grades Regulares* para uma distribuição esparsa dos objetos.

TAB. 6.6: Média de pares gerados pelo método *Grades Regulares* - distribuição esparsa

	100	**300**	**500**	**1000**
GR4x4x4	**967,7**	**9.219,1**	**25.922,1**	**99.695,1**
GR2x2x2	1.649,9	14.754,1	41.387,1	159.025,7
GR2x2x1	2.332,5	21.392,3	60.726,1	233.714,4
GR1x2x1	3.433,5	30.911,6	87.380,3	342.442,6

Para este cenário a configuração que mostrou-se mais eficiente quanto a redução de pares gerados foi a grade definida com 4 células para cada eixo (GR4x4x4). No gráfico da Figura 6.20 pode ser observado mais detalhadamente os pares gerados a partir de 100 objetos em cada passo de simulação.

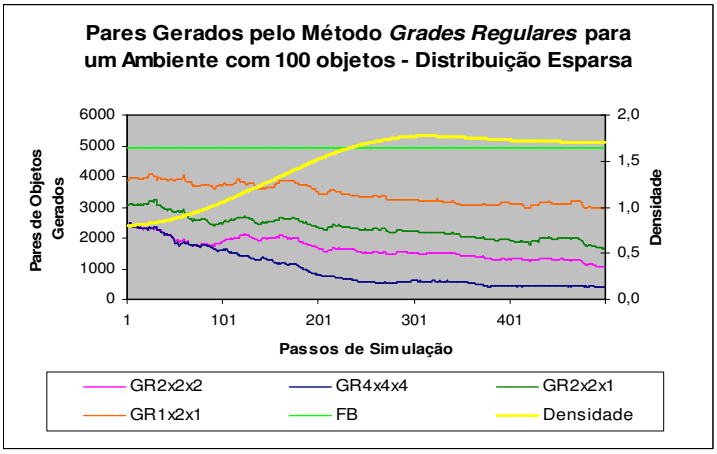


FIG. 6.20: Número de pares gerados pelo método *Grades Regulares* para uma distribuição esparsa com 100 objetos.

Para um ambiente esparso, quanto mais células for dividida uma grade, menor será o número de pares gerados. Isso se deve ao fato de não haverem muitos objetos concentrados em uma mesma célula. Por isso, a configuração GR4x4x4 se mostrou a mais eficiente na redução do número de pares gerados se comparada às demais configurações testadas.

É interessante notar no gráfico da Figura 6.20 que nos primeiros passos de simulação as configurações GR2x2x2 e GR4x4x4 geram praticamente a mesma quantidade de pares. Isso se deve ao fato dos objetos do ambiente ainda não se encontram muito dispersos uns dos outros, o que acaba ocasionando o mesmo comportamento para essas configurações. As configurações GR2x2x1 e GR1x2x1 não foram eficientes na redução de pares devido a possuírem um número menor de células.

Independente da configuração adotada, os testes mostraram que para este método, quando aplicado a uma distribuição esparsa de objetos, a quantidade de pares gerados é

inferior à quantidade gerada pela *Força Bruta*.

6.5.3 MÉTODO *BSP-TREE*

O gráfico da Figura 6.21 mostra a variação das médias de pares gerados para as configurações do método *BSP-Tree* no que se refere a profundidade da árvore criada para o ambiente de simulação com 100, 300, 500 e 1000 Objetos. A Tabela 6.7 apresenta os valores das médias obtidas para cada configuração.

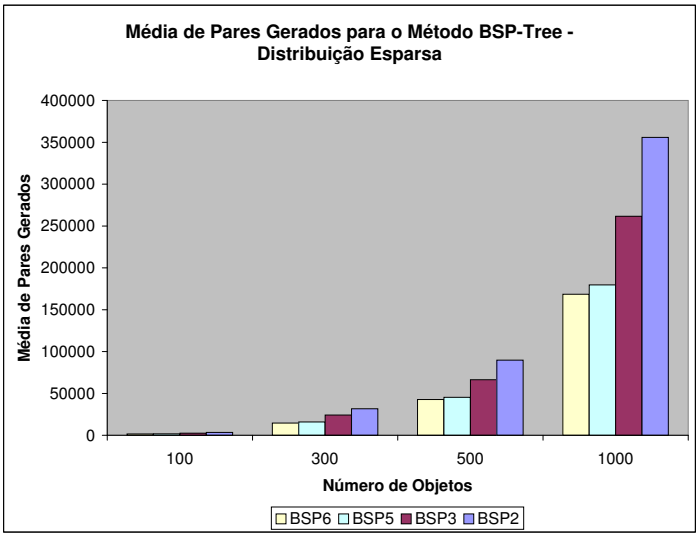


FIG. 6.21: Média de pares gerados pelo método *BSP-Tree* para uma distribuição esparsa dos objetos.

TAB. 6.7: Média de pares gerados pelo método *BSP-Tree* - distribuição esparsa

	100	**300**	**500**	**1000**
BSP6	**1.543,2**	**14.574,6**	**42.647,2**	**168.478,9**
BSP5	1.677,1	15.861,2	45.307,1	179.583,5
BSP3	2.478,2	24.147,5	66.318,4	261.670,3
BSP2	3.368,9	31.692,9	89.742,9	355.898,3

Para este cenário a configuração que mostrou-se mais eficiente quanto a redução de pares gerados foi a profundidade 6 (BSP6). No gráfico da Figura 6.22 pode ser observado mais detalhadamente os pares gerados a partir de 100 objetos em cada passo de simulação.

Quanto aos resultados obtidos dos testes realizados com o método *BSP-Tree* pode-se concluir que quanto maior a profundidade adotada, menos pares de objetos serão gerados, visto que a cada plano de divisão criado as regiões definidas tendem a ser menores. Com

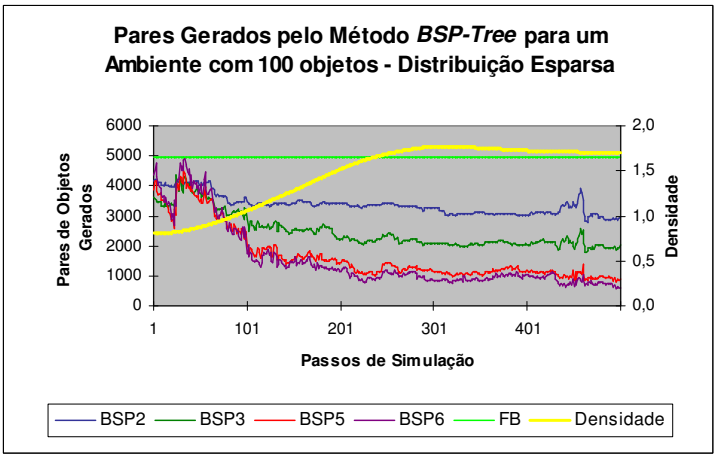


FIG. 6.22: Número de pares gerados pelo método *BSP-Tree* para uma distribuição esparsa com 100 objetos.

isso, quanto mais dispersos os objetos se encontrarem, cada região do ambiente provavelmente irá considerar um número menor de objetos para serem processados. Desse modo, a profundidade 6 mostrou-se mais eficiente na redução de pares, tendo a profundidade 5 resultados muito próximos.

É interessante notar no gráfico da Figura 6.22 que nos primeiros passos de simulação, o número de pares gerados para as configurações testadas são bastantes equivalentes. Neste caso, pode-se concluir que para as profundidades 5 e 6 houveram duplicações de objetos, de modo que apresentaram praticamente a mesma redução alcançada pelas profundidades 2 e 3. No entanto, no decorrer da simulação, à medida que os objetos vão se afastando um dos outros as profundidades 2 e 3 não se mostram tão eficientes devido a cada uma das regiões obtidas pelos planos de divisão conterem muitos objetos.

Para este método também os testes mostraram que a quantidade de pares gerados é inferior à quantidade gerada pela *Força Bruta*.

6.5.4 MÉTODO *SWEEP AND PRUNE*

O gráfico da Figura 6.23 mostra a variação das médias de pares gerados para as configurações do método *Sweep and Prune* no que se refere a quantidade de eixos escolhidos para varredura do ambiente de simulação com 100, 300, 500 e 1000 Objetos. A Tabela 6.8 apresenta os valores das médias obtidas para cada configuração.

Para este cenário a configuração que mostrou-se mais eficiente quanto a redução de pares gerados foi a que levou em consideração os 3 eixos coordenados (SaP3). No gráfico

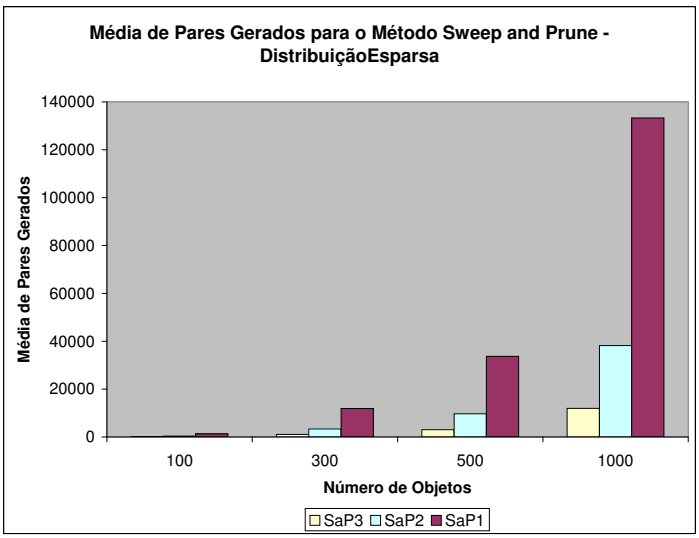


FIG. 6.23: Média de pares gerados pelo método *Sweep and Prune* para uma distribuição esparsa dos objetos.

TAB. 6.8: Média de pares gerados pelo método *Sweep and Prune* - distribuição esparsa

	100	**300**	**500**	**1000**
SaP3	**111,2**	**1.061,7**	**3.021,9**	**11.930,6**
SaP2	350,1	3.337,5	9.719,1	38.208,9
SaP1	1.277,1	11.899,3	33.697,8	133.277,4

da Figura 6.24 pode ser observado mais detalhadamente os pares gerados a partir de 100 objetos em cada passo de simulação.

Assim como para a distribuição densa dos objetos, o método *Sweep and Prune* configurado para utilizar 3 eixos para varredura foi mais eficiente na redução de pares, como era o esperado. Considerando 1 ou 2 eixos resultou em uma quantidade maior de pares.

Como neste tipo de cenário os objetos encontram-se mais dispersos, a quantidade de objetos que possuem suas projeções em sobreposição é menor, e dessa forma a geração de pares é menor para este método. Essa tendência é mostrada na Figura 6.24, onde pode-se notar também que a quantidade de pares gerados é bem menor que a obtida pela *Força Bruta*.

6.5.5 MÉTODO *AOIP*

A Figura 6.25 mostra a variação das médias de pares gerados para as configurações do método *AoIP* no que se refere ao tamanho da área considerada para o ambiente de simulação com 100, 300, 500 e 1000 Objetos. A Tabela 6.9 apresenta os valores das médias

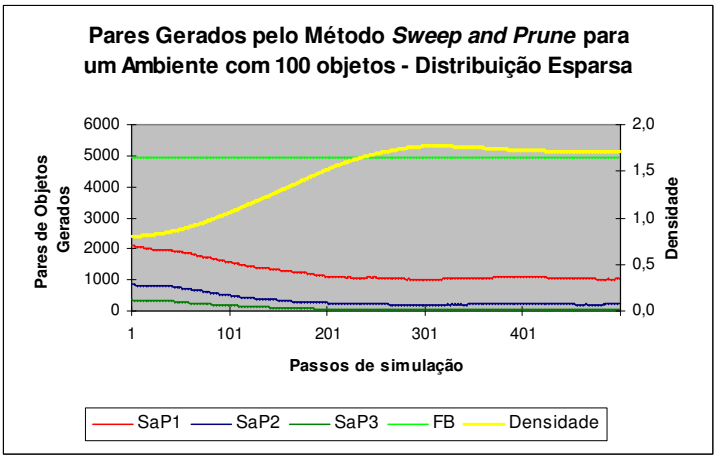


FIG. 6.24: Número de pares gerados pelo método *Sweep and Prune* para uma distribuição esparsa com 100 objetos.

obtidas para cada configuração.

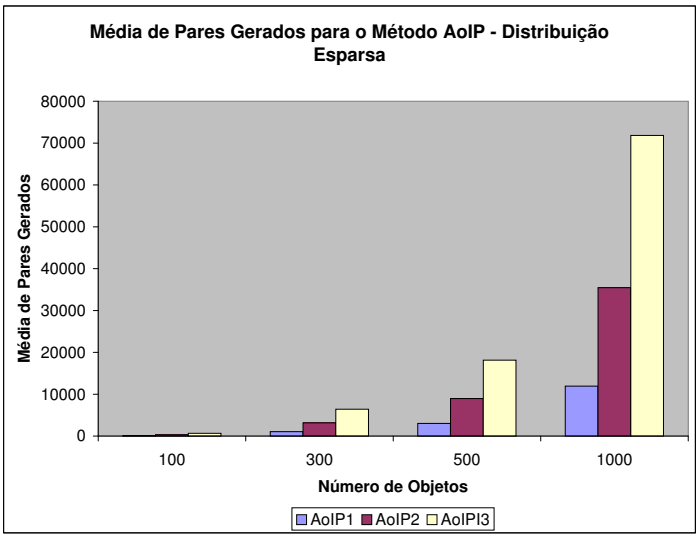


FIG. 6.25: Média de pares gerados pelo método *AoIP* para uma distribuição esparsa dos objetos.

Para este cenário a configuração que mostrou-se mais eficiente quanto a redução de pares gerados foi a que considera uma área de interesse de tamanho igual ao tamanho do objeto (AoIP1). No gráfico da Figura 6.26 pode ser observado mais detalhadamente os pares gerados a partir de 100 objetos em cada passo de simulação.

Em ambientes com objetos esparsos a probabiliadade de muitos objetos estarem ao alcance de uma área de interesse é menor, sendo que aumentando-se a área esta probabi-

TAB. 6.9: Média de pares gerados pelo método *AoIP* - distribuição esparsa

	100	**300**	**500**	**1000**
AoIP1	**111,2**	**1.061,7**	**3.021,9**	**11.930,6**
AoIP2	322,3	3.148,4	8.951,1	35.453,9
AoIP3	650,3	6.413,2	18.146,3	71.819,9

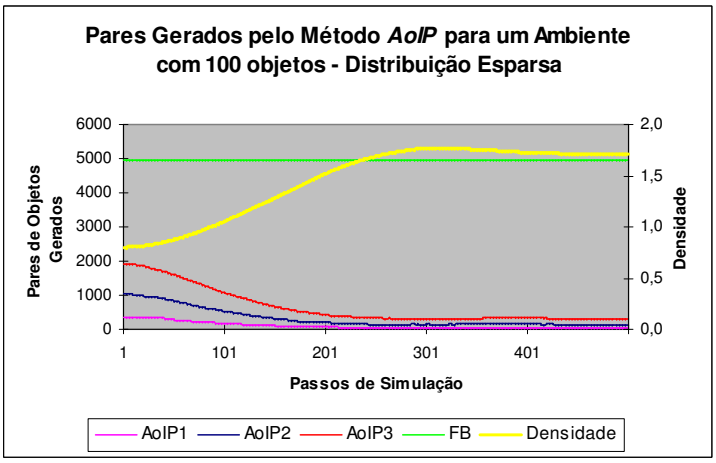


FIG. 6.26: Número de pares gerados pelo método *AoIP* para uma distribuição esparsa com 100 objetos.

lidade aumenta. O gráfico da Figura 6.26 mostra essa tendência, como também, mostra que para as configurações adotas para a calibração da área de interesse a quantidade de pares gerados é bem menor que a obtida pela *Força Bruta*. No entanto, se pelo menos uma das áreas fosse do mesmo tamanho do ambiente, o resultado obtido seria igual ao da *Força Bruta*, pois todos os objetos do ambiente encontrariam-se inseridos nesta área.

6.5.6 CONSIDERAÇÕES SOBRE OS RESULTADOS

O gráfico da Figura 6.27 mostra a variação das médias de pares gerados para os métodos testados e suas configurações.

Como pode ser observado, para este tipo de ambiente a redução de pares alcançada por todos métodos de acordo com as configurações testadas foi inferior à quantidade de pares gerada pela *Força Bruta*. Isso se deve à duplicação de objetos não ter sido significativa neste tipo de ambiente ao ponto de influenciar na quantidade de pares obtidos, como ocorreu para o ambiente denso.

Os melhores resultados quanto a redução do número de pares de objetos a serem considerados nos testes de colisão foram obtidos pelos métodos *AoIP* e *Sweep and Prune*. Além desses métodos, outro que se mostrou eficiente foi *Grades Regulares* com configuração

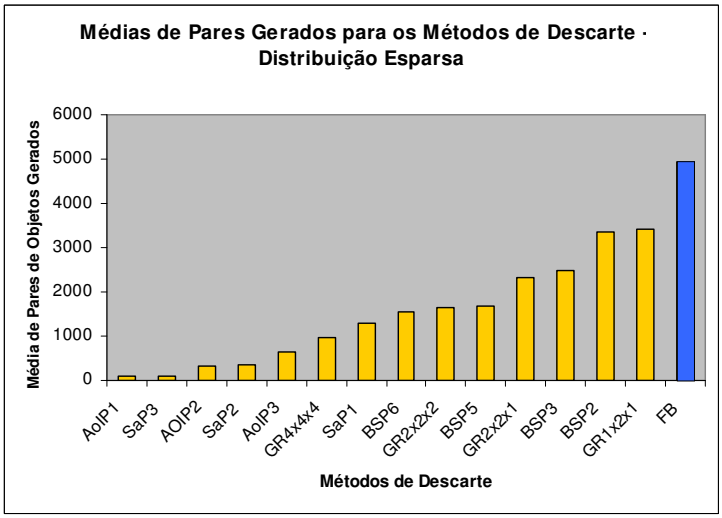


FIG. 6.27: Média de pares gerados pelos métodos de descarte para uma distribuição esparsa com 100 objetos.

4x4x4, devido a este tipo de particionamento ser indicado para ambientes que tem uma distribuição dos objetos mais uniforme. O método *BSP-Tree* com profundidade 6 também foi um dos que descartou muitos pares.

Os métodos *Sweep and Prune* e *AoIP* resultaram no mesmo comportamento descrito para ambientes densos. O gráfico da Figura 6.28 mostra o comportamento desses e dos demais métodos.

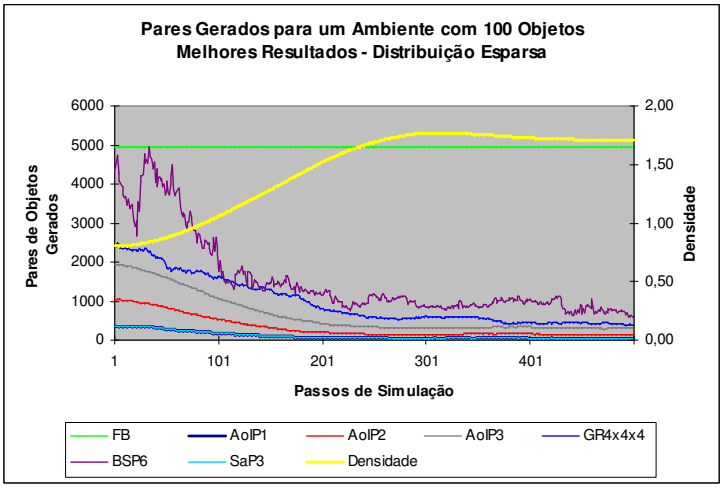


FIG. 6.28: Melhores resultados obtidos para uma distribuição esparsa com 100 objetos.

Para os métodos *AoIP* e *Sweep and Prune* quanto mais dispersos encontraram-se os objetos do ambiente, mais foi reduzido o número de pares resultantes, como pode ser visto

no gráfico da Figura 6.29.

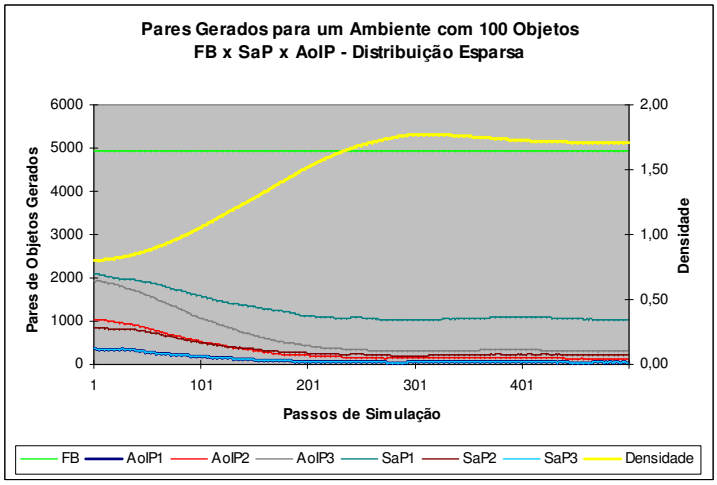


FIG. 6.29: Comparação entre os resultados obtidos para os métodos *AoIP* e *Sweep and Prune* para uma distribuição esparsa com 100 objetos.

Assim como ocorreu para ambientes densos, os métodos *AoIP* e *Sweep and Prune* geraram a mesma quantidade de pares quando considerada uma área de interesse de mesmo tamanho do objeto e considerado 3 eixos, respectivamente.

6.6 TEMPOS DE PROCESSAMENTO OBTIDOS NA DISTRIBUIÇÃO ESPARSA

Nesta seção serão apresentados os tempos médios de processamentos obtidos na Fase de Descarte para cada método simulado em uma distribuição esparsa com 100, 300, 500 e 1000 objetos. Os resultados gráficos obtidos, assim como para a Seção 6.4, são apresentados no Apêndice 9.3. A seguir é discutido o comportamento de cada método de descarte simulado em um ambiente esparso.

6.6.1 MÉTODO *FORÇA BRUTA*

Assim como ocorre para uma distribuição densa de objetos, o tempo de processamento obtido para a *Força Bruta* em um ambiente esparso é proveniente da etapa de geração de pares de objetos. Desse modo, quanto maior a quantidade de objetos, maior será o tempo de processamento desse método.

6.6.2 MÉTODO *GRADES REGULARES*

Grades Regulares aplicadas em uma ambiente esparso possuem para a etapa de criação praticamente os mesmos tempos de processamento obtidos para um ambiente denso. Esse método consome mais tempo nas etapas de atualização e geração de pares.

De uma modo geral, observou-se que, como ocorre para um ambiente denso, quanto maior o número de células na grade, maior foi o tempo de atualização. No entanto, para a etapa de geração de pares, pode-se observar que as configurações com um maior número de células tiveram seu tempo de processamento reduzido. Apesar de existirem mais divisões do ambiente, em um ambiente esparso a quantidade de objetos que cada célula processa acaba sendo muito menor, e consequentemente, o tempo de processamento é reduzido.

6.6.3 MÉTODO *BSP-TREE*

O Método *BSP-Tree* aplicado a um ambiente esparso também gerou tempos de processamento para a etapa de criação relativamente baixos.

Neste caso, a etapa de atualização também foi a que consumiu a maior quantidade de processamento. Independente da distribuição espacial dos objetos no ambiente, reconstruir a árvore da BSP-Tree a cada passo de simulação é um processo custoso computacionalmente, principalmente se for levando em consideração que os planos de divisão devem ser recalculados.

No tempo de processamento gerado pela etapa de geração de pares observou-se que apesar de ainda existir o tempo gasto no percurso da árvore até seus nós-folhas, o tempo de processamento de objetos que encontram-se dentro das células acaba sendo menor para uma ambiente esparso. Isso faz com que de um modo geral, o tempo de processamento da etapa de geração de pares seja reduzido.

6.6.4 MÉTODO *SWEEP AND PRUNE*

O comportamento do método *Sweep and Prune* para uma distribuição esparsa de objetos é basicamente o mesmo obtido para ambientes densos. A etapa de geração de pares é a que consome mais tempo de processamento.

Para este tipo de distribuição espacial, também foram obtidos os maiores tempos de processamento pela configuração que utiliza 3 eixos para varredura, seguida pela que considera 2 eixos e a que considera 1 eixo.

6.6.5 MÉTODO *AOIP*

A etapa de criação para o Método *AoIP* também apresentou valores relativamente baixos. Observou-se que da mesma forma que para um ambiente denso, a etapa de atualização e a etapa de geração de pares consumiram grande parte do processamento total. Neste caso, quanto menor o tamanho da área de interesse menor o tempo de processamento obtido.

6.6.6 CONSIDERAÇÕES SOBRE OS RESULTADOS

Os gráficos da Figura 6.30 apresentam os tempos médios de processamento e a média de pares de objetos gerados pelos métodos de descarte. Também neste caso, pode-se verificar que não há relação entre a quantidade de pares gerados e o tempo de processamento gasto pela fase de descarte.

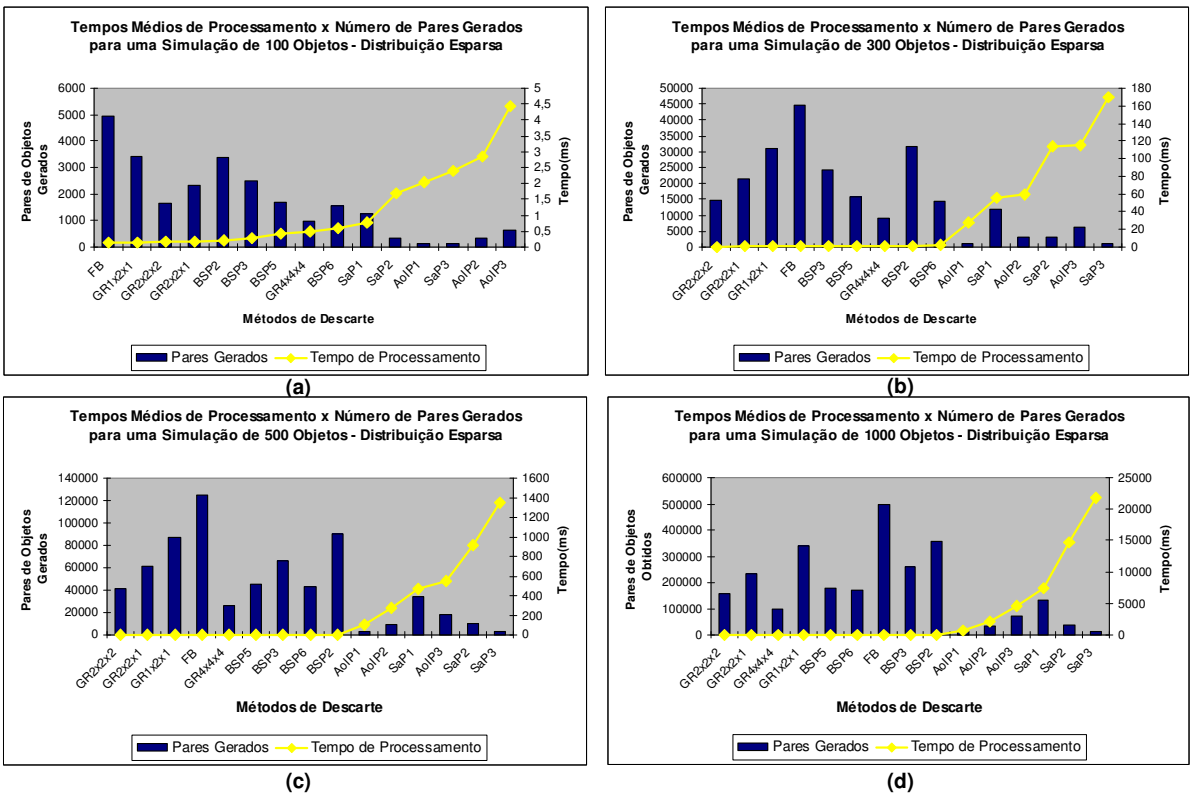


FIG. 6.30: Tempos médios de processamento e média do número de pares gerados para uma distribuição esparsa: (a) valores obtidos para 100 objetos, (b) valores obtidos para 300 objetos, (c) valores obtidos para 500 objetos e (d) valores obtidos para 1000 objetos.

O comportamento apresentado para os métodos de descarte para um ambiente esparso

foi semelhante ao apresentado para um ambiente denso. Os métodos *AoIP* e *Sweep and Prune* apresentaram os maiores tempos médios de processamento para a fase de descarte, sendo que o método *Sweep and Prune* com varredura em 3 eixos (SaP3) teve o maior tempo e os métodos *Grades Regulares* e a *Força Bruta* foram os que apresentaram os menores tempos de processamento.

Como era de se esperar, o método *AoIP* com uma área de tamanho igual ao do objeto e o *Sweep and Prune* com 3 eixos de varredura, foram os métodos que tiveram os menores tempos de processamento. No entanto, o método *Sweep and Prune* apresentou uma diferença significativa quanto ao tempo de processamento se comparado ao método *AoIP*. O método *Sweep and Prune* apesar de gerar a mesma quantidade de pares de objetos que o método *AoIP*, possui um tempo de processamento maior.

7 CONCLUSÕES E TRABALHOS FUTUROS

7.1 CONSIDERAÇÕES FINAIS

Para evitar testes de intersecção entre todos os objetos de um ambiente virtual, torna-se necessário descartar comparações entre objetos que encontram-se distantes uns dos outros. Neste contexto, a fase de descarte identifica os pares de objetos que realmente tem probabilidade de colidirem por estarem próximos no ambiente virtual.

A quantidade de pares gerados pela fase de descarte tem influência direta na quantidade de tempo de processamento da fase de refinamento. Quanto mais pares forem gerados pela fase de descarte, mais será custoso o processamento da fase de refinamento. Neste trabalho foi proposto um método de descarte utilizando o conceito de áreas de interesse de modo a acelerar a fase de refinamento.

A partir dos experimentos realizados foi possível verificar que a utilização de áreas de interesse na fase de descarte reduziu consideravelmente a quantidade de objetos a serem considerados nos testes de colisão. Isso nos leva às seguintes conclusões:

- Com a utilização de áreas de interesse obtem-se um particionamento vinculado a localização de cada objeto e não ao ambiente como um todo, como acontece com os métodos de particionamento espacial apresentados, aos quais subdividem o ambiente virtual em partições fixas e estáticas;
- Um objeto possui uma visão do mundo de acordo com os objetos que estão dentro do seu domínio, o qual é estipulado pela extensão da sua área de interesse;
- Cada objeto passa a gerenciar a sua área de interesse e, consequentemente, os seus testes de colisão.
- O fato de uma área de interesse estar vinculada a um objeto dinâmico faz com que tenha mobilidade pelo ambiente. Essa mobilidade reduz a probabilidade de haver um número muito grande de objetos em uma mesma área de interesse, mesmo quando estes objetos encontram-se concentrados em uma mesma região;
- Um problema comum nos métodos de particionamento é gerenciar a migração de objetos entre as regiões ou ainda o fato de um objeto se encontrar na fronteira entre duas regiões. Com a utilização de áreas de interesse não há fronteiras, visto que cada

área de interesse é independente uma da outra e acompanha um objeto específico. Neste caso não existe uma região fixa do ambiente como acontece nas técnicas de particionamento espacial.

- O comportamento obtido com a utilização do método *AOIP* em ambientes densos e ambientes esparsos foi bastante semelhante, conseguindo reduzir significativamente os pares de objetos em comparação aos outros métodos testados. Apesar de descartar muitos pares de objetos, o método apresentou um elevado tempo de processamento com relação a atualização de suas estruturas de dados e o processo para a geração de pares. No entanto, acredita-se que esse tempo de processamento pode ser compensado pela redução do números de polígonos a serem comparados pela fase de refinamento.
- Observou-se que quanto menor o tamanho definido para uma área de interesse menor a quantidade de pares gerados e menor o processamento necessário para obter estes pares. Desse modo, utilizar áreas de interesse para objetos rígidos com tamanhos maiores que as dimensões do objeto pode não ser vantajoso. Áreas de Interesse maiores podem fazer sentido para objetos deformáveis ou articulados que constantemente estão mudando de tamanho. Uma área de interesse que pudesse englobar todas as possíveis animações de um objeto seria capaz de prever colisões futuras.

7.2 TRABALHOS FUTUROS

Como trabalhos futuros propõem-se os seguintes tópicos:

- Analisar o comportamento do método *AoIP* quanto aos seguintes aspectos: utilização de outros tipos dos objetos (estáticos, articulados, deformáveis), variação do tamanho dos objetos e variação na velocidade dos objetos.

- Buscar por mecanismos que venham a reduzir o tempo de atualização do método de modo que as estruturas que armazenam os objetos que encontram-se dentro das áreas de interesse não necessitem serem recriadas constantemente.

8 REFERÊNCIAS BIBLIOGRÁFICAS

AGARWAL, P., RAJAGOPALAN, S. e PRABHAKARAN, B. Minimizing probable collision pairs searched in interactive animation authoring. *The Visual Computer*, 24(5): 347–359, 2008.

BARAFF, D. *Dynamic Simulation of Non-Penetrating Rigid Bodies*. Tese de Doutorado, Cornell University, 1992.

BENFORD, S. e FAHLEN, L. E. A spatial model of interaction in large virtual environments. *Third European Conference on Computer Supported Cooperative Work*, págs. 107–123, 1993.

BENTLEY, J. L. Multidimensional binary search trees used for associative searching. *Communication on the ACM*, 18(9):509–517, 1975.

BERGEN, G. V. D. *Collision Detection in Interactive 3D Environments*. Morgan Kaufmann, 2004. ISBN 155860801X.

COHEN, M. e MANSSOUR, I. H. *OpenGL - Uma Abordagem Prática e Objetiva*. Novatec, 2006.

ERICSON, C. *Real-Time Collision Detection*. Morgan Kaufmann, 2005. ISBN 1558607323.

FARES, C. e HAMAN, Y. Collision detection for rigid bodies: A state of the art review. *Computer Graphics and Applications (GraphiCon'2005), Fifteenth International Conference*, 2005.

FUCHS, H., KEDEM, Z. M. e NAYLOR, B. F. On visible surface generation by a priori tree structures. *ACM Computer Graphics*, 1980.

GREENHALGH, C. *Large Scale Collaborative Virtual Environments*. Tese de Doutorado, Computer Science Department, University of Nottingham, UK., 1997. URL `http://www.crg.cs.nott.ac.uk/ cmg/cgreenhalgh-thesis-singlespaced.pdf`.

JIMÉNEZ, P., THOMAS, F. e TORRAS, C. 3d collision detection: A survey. *Computers and Graphics*, 25:269–285, 2001.

KIM, G. J. *Designing Virtual Reality Systems*. Springer-Verlag, 2005.

LIN, M. e GOTTSCHALK, S. Collision detection between geometric models: A survey. *In Proceedings of IMA Conference on Mathematics of Surfaces,*, 1:602–608, 1998.

LUQUE, R. G.; COMBA, L. D. e FREITAS, C. M. D. S. Broad-phase collision detection using semi-adjusting bsp-trees. Em *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, págs. 179–186, New York, NY, USA, 2005. ACM. ISBN 1-59593-013-2.

MACEDONIA, M., ZYDA, M., PRATT, D., BARHAM, P. e ZESWITZ, S. Npsnet: A network software architecture for large scale virtual environments. *Presence*, 3(4): 265–287, 1994.

MALFATTI, S., SANTOS, S. R., FRAGA, L. M., JUSTEL, C. e OLIVEIRA, J. C. Encima: A graphics engine for the development of multimedia and virtual reality applications. *X Symposium on Virtual and Augmented Reality (SVR 2008)*, págs. 313–321, 2008.

MOLLER, T. e HAINES, E. *Real-Time Rendering.* A K Peters, 1971.

MOORE, M. e WILHELMS, J. Collision detection and response for computer animation. Em *Computer Graphics*, págs. 289–298, 1988.

MORSE, K. L. Interest management in large-scale distributed simulations. Technical report, Report ICS-TR-96-27, UC Irvine, 1996.

OLIVEIRA, J. C. e GEORGANAS, N. D. Velvet: an adaptive hybrid architecture for very large virtual environments. *Presence: Teleoperators Virtual Environment*, 12(6): 555–580, 2003. ISSN 1054-7460.

ROCHA, R. e RODRIGUES, M. Análise de desempenho de algoritmos para detecção de colisão em ambientes gráficos interativos. *Proceedings of the XXVII Congresso da SBC, Concurso de Trabalhos de Iniciação Científica (CTIC)*, 2007.

SAMET, H. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, MA, 1990.

SANTOS, S. R. D., FRAGA, L. M., TRENHAGO, P. R., DE OLIVEIRA, J. C. e MALFATTI, S. M. Using a rendering engine to support the development of immersive virtual reality applications. Em *IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems (VECIMS2008)*, págs. 74–79, Istanbul, Turquia, 2008.

VAN HOOK, D. J.; RAK, S. J. e CALVIN, J. O. Approaches to relevance filtering. *11th Workshop on Standards for the Interoperability of Distributed Simulations*, págs. 367–369, 1994.

9 APÊNDICES

9.1 APÊNDICE 1: DENSIDADES PARA OS EXPERIMENTOS SIMULADOS COM 300, 500 E 1000 OBJETOS

Neste apêndice são apresentadas as densidades calculadas para os dois tipos de cenários descritos na Seção 6.2 do Capítulo 6 simulados com 300, 500 e 1000 objetos.

A densidade calculada para um tipo de cenário é uma medida relativa, não sendo dependente da variação da quantidade de objetos. As densidades calculadas para cada cenário indicam quão próximos os objetos estão uns dos outros, não havendo relação das mesmas entre si.

Observar que as densidades calculadas para cada variação da quantidade de objetos no ambiente apresentam-se com valores próximos em cada tipo de cenário. Isso acontece pois em todas as simulações os objetos dinâmicos são gerados no centro do ambiente e se movimentam em direção das paredes.

A Figura 9.1 mostra as densidades obtidas para a execução de 300, 500 e 1000 objetos em 500 passos de simulação.

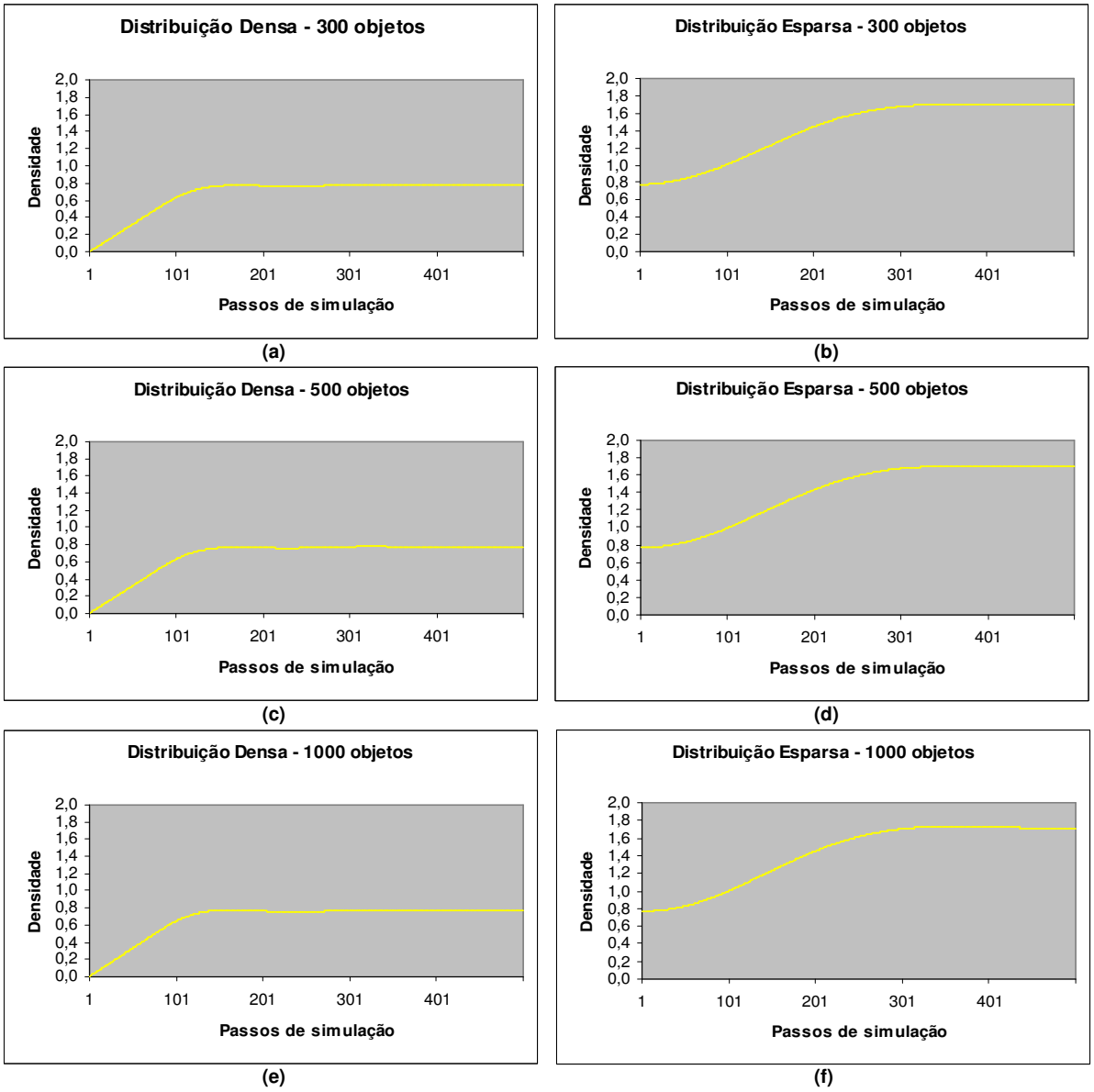


FIG. 9.1: Densidades dos cenários simulados: em (a) e (b) para 300 objetos; em (c) e (d) para 500 objetos e em (e) e (f) para 1000 objetos.

9.2 APÊNDICE 2: RESULTADOS PARA A GERAÇÃO DE PARES

Neste apêndice são apresentados os resultados obtidos pelos métodos implementados para a geração de pares considerando os cenários simulados com 300, 500 e 1000 Objetos.

O gráfico da Figura 9.2 mostra as médias de pares gerados pelos métodos de descarte implementados em cada tipo de distribuição espacial adotada: em (a) e (b) resultados obtidos para 300 objetos; em (c) e (d) resultados obtidos para 500 objetos e em (e) e (f) resultados obtidos para 1000 objetos.

Nos gráficos das Figuras 9.3, 9.4, 9.5, 9.6 e Figura 9.7 são apresentados em detalhe a quantidade de pares gerados em cada passo de simulação para cada tipo de distribuição espacial adotada: em (a) e (b) resultados obtidos para 300 objetos; em (c) e (d) resultados obtidos para 500 objetos e em (e) e (f) resultados obtidos para 1000 objetos.

Os gráficos da Figura 9.8 mostram uma comparação dos melhores resultados obtidos em cada passo de simulação para cada tipo de distribuição espacial adotada: em (a) e (b) resultados obtidos para 300 objetos; em (c) e (d) resultados obtidos para 500 objetos e em (e) e (f) resultados obtidos para 1000 objetos.

Por fim, nos gráficos da Figura 9.9 são apresentadas comparações dos resultados obtidos pelos métodos que obtiveram a maior redução na quantidade de pares gerados na simulação, *AoIP* e *Sweep and Prune*, para cada tipo de distribuição espacial adotada: em (a) e (b) resultados obtidos para 300 objetos; em (c) e (d) resultados obtidos para 500 objetos e em (e) e (f) resultados obtidos para 1000 objetos.

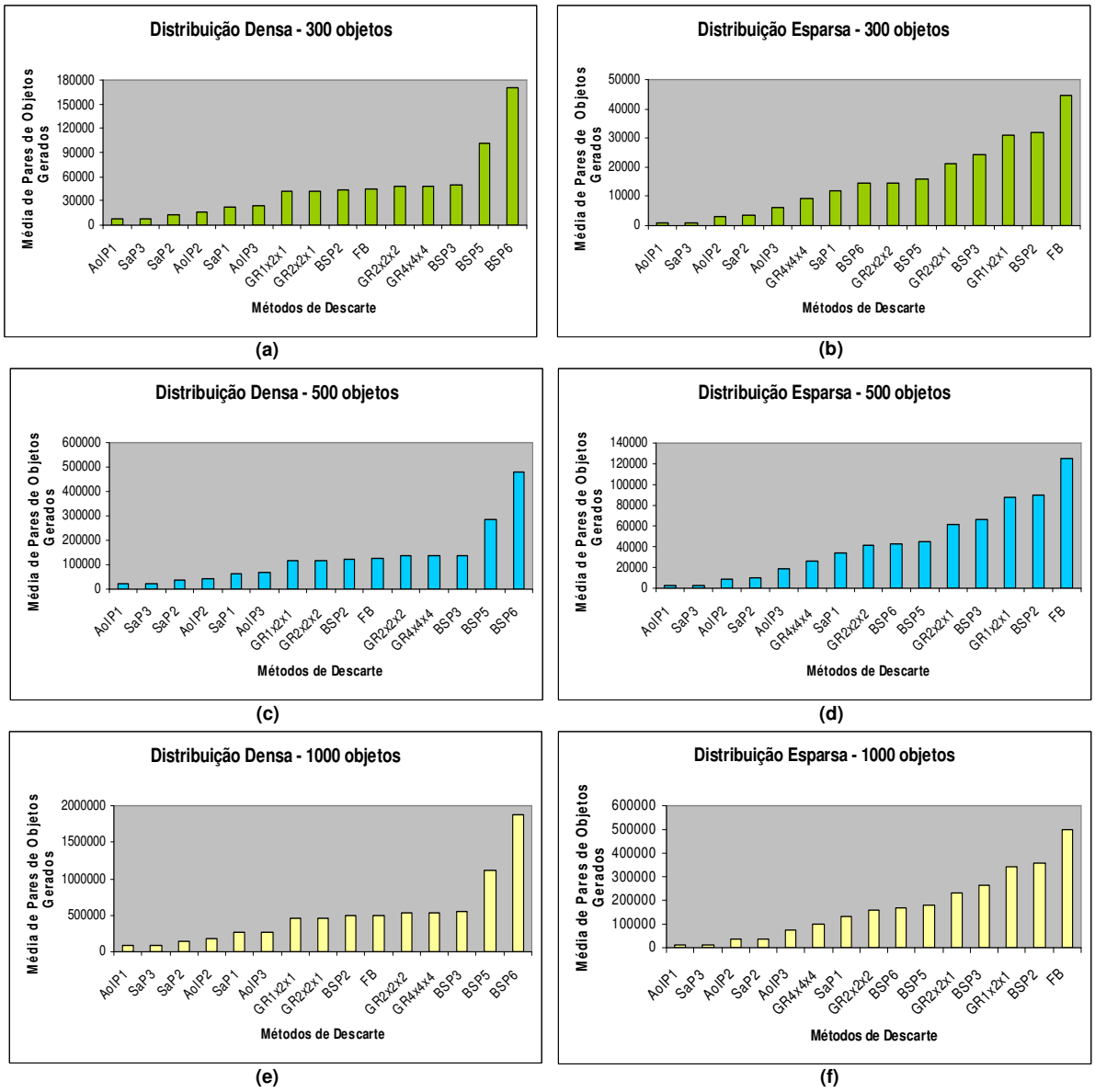


FIG. 9.2: Média de pares gerados pelos métodos de descarte para uma distribuição densa e esparsa de 300, 500 e 1000 objetos.

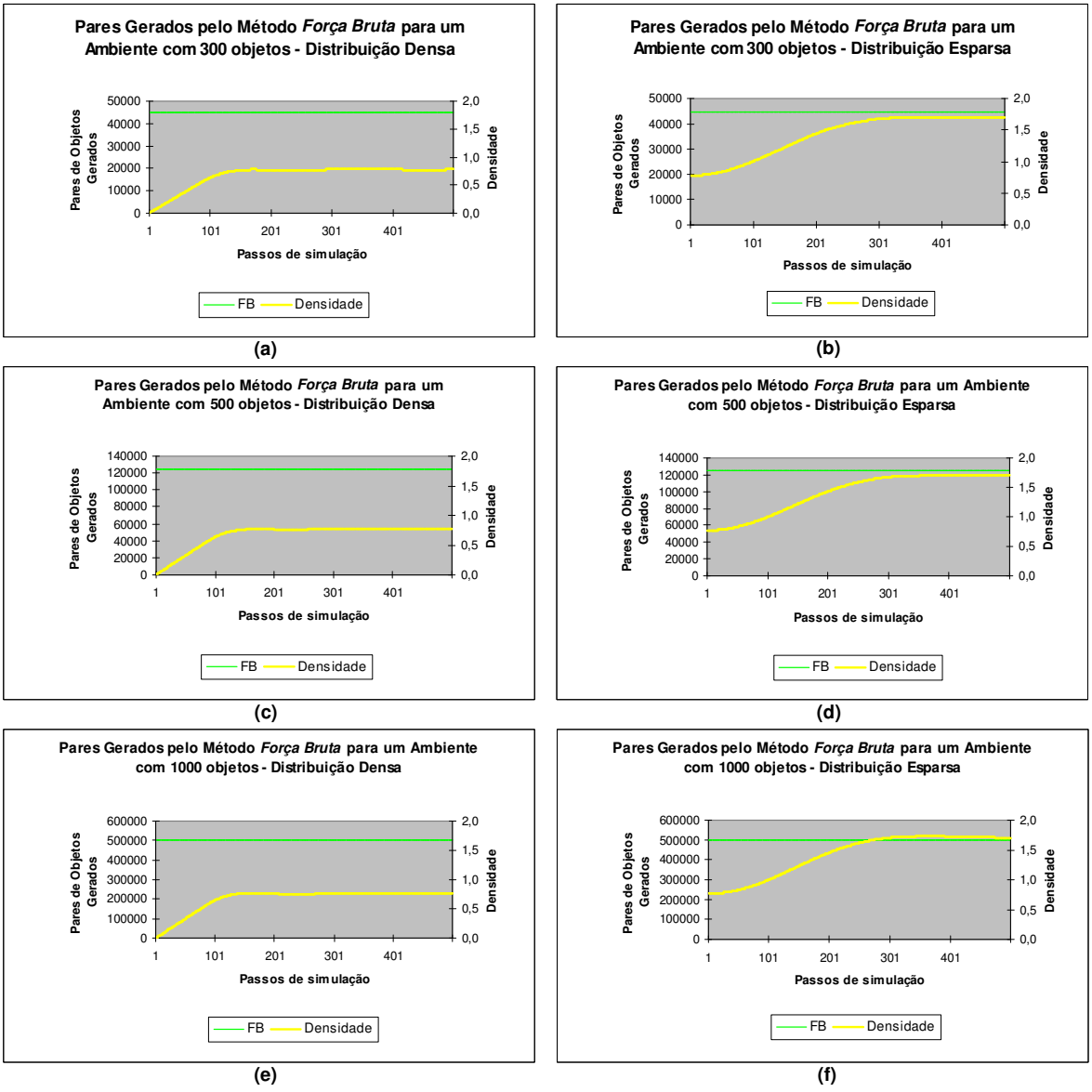


FIG. 9.3: Número de pares gerados pelo método *Força Bruta* para uma distribuição densa e esparsa com 300, 500 e 1000 objetos.

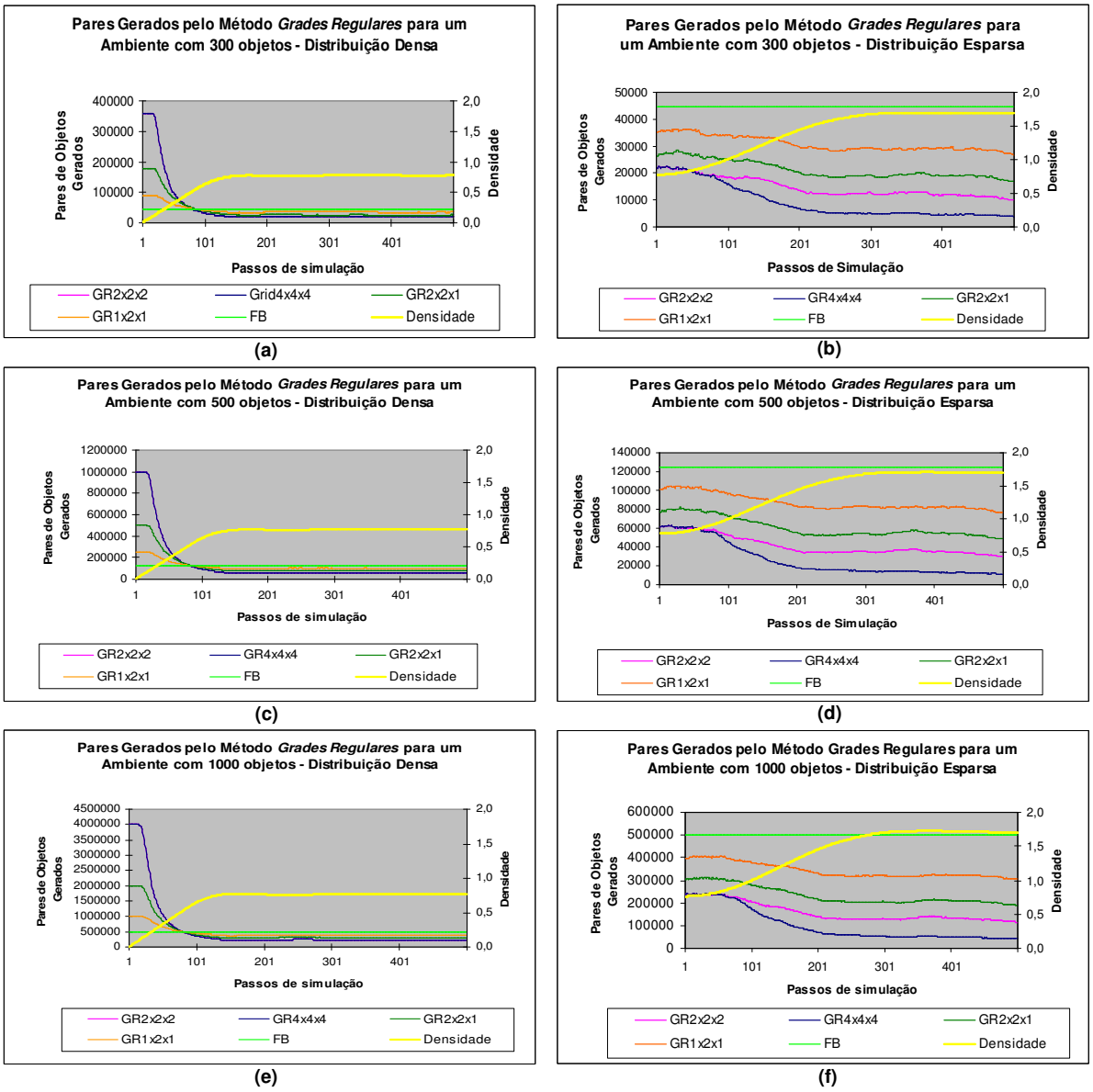


FIG. 9.4: Número de pares gerados pelo método *Grades Regulares* para uma distribuição densa e esparsa com 300, 500 e 1000 objetos.

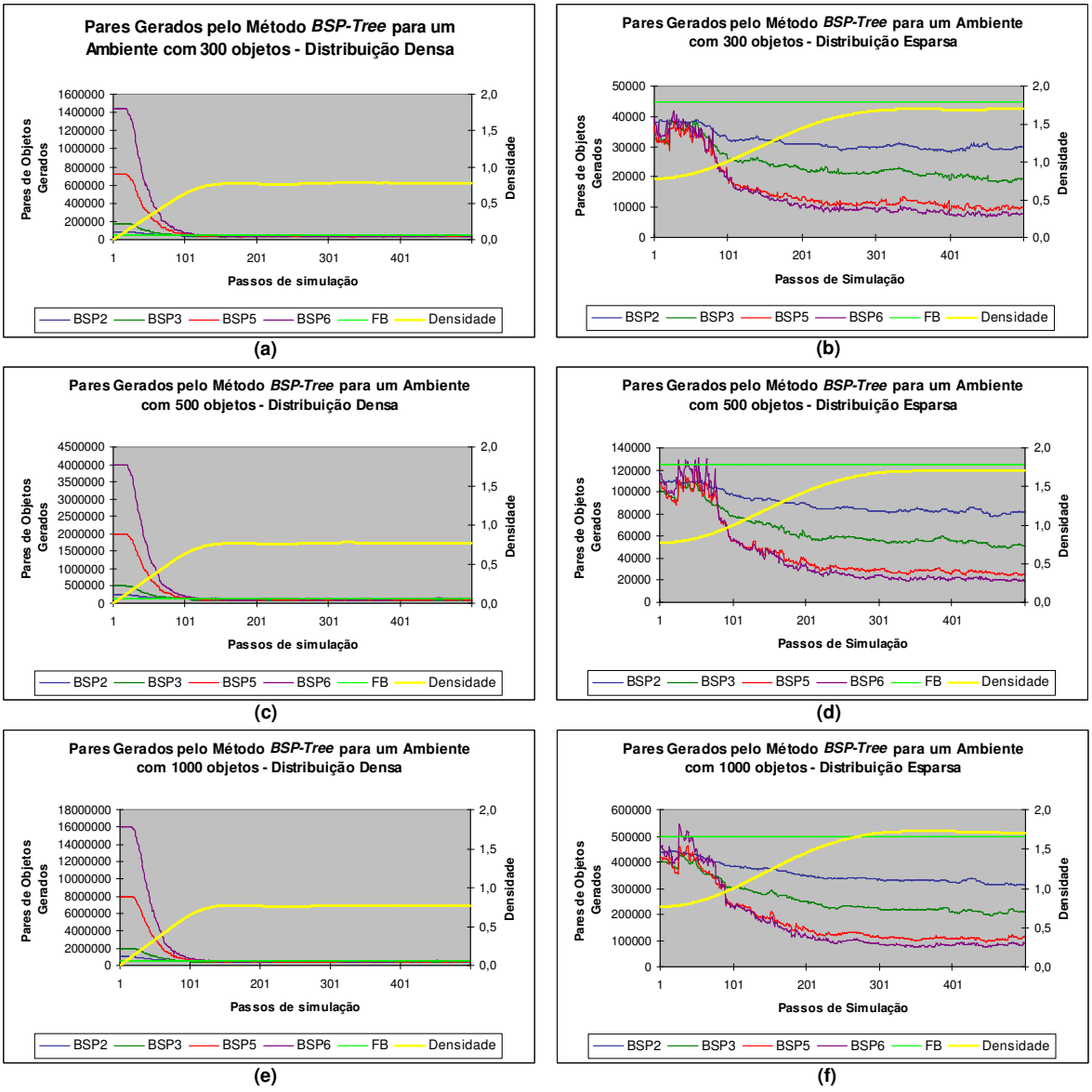


FIG. 9.5: Número de pares gerados pelo método *BSP-Tree* para uma distribuição densa e esparsa com 300, 500 e 1000 objetos.

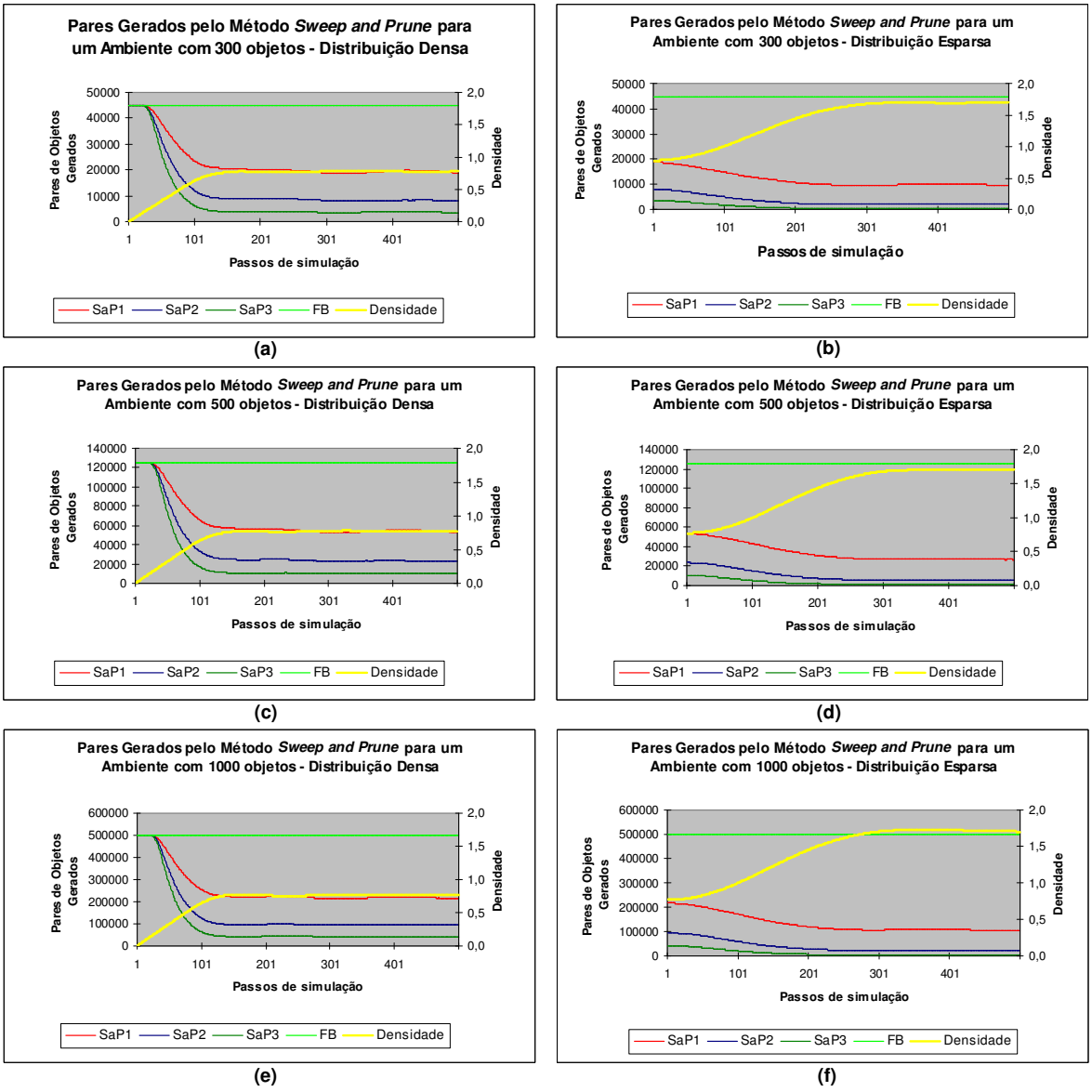


FIG. 9.6: Número de pares gerados pelo método *Sweep and Prune* para uma distribuição densa e esparsa com 300, 500 e 1000 objetos.

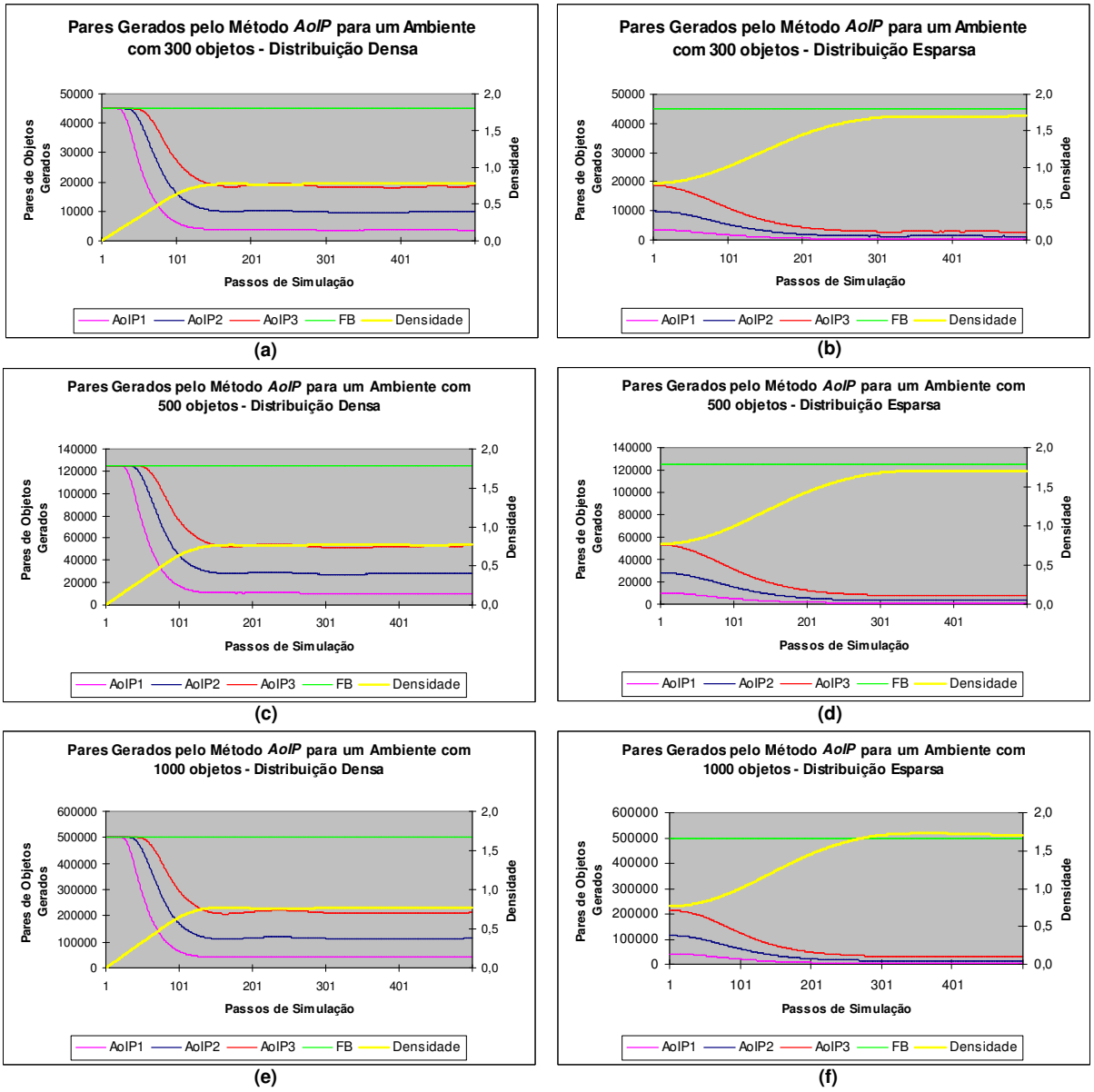


FIG. 9.7: Número de pares gerados pelo método *AoIP* para uma distribuição densa e esparsa de 300, 500 e 1000 objetos.

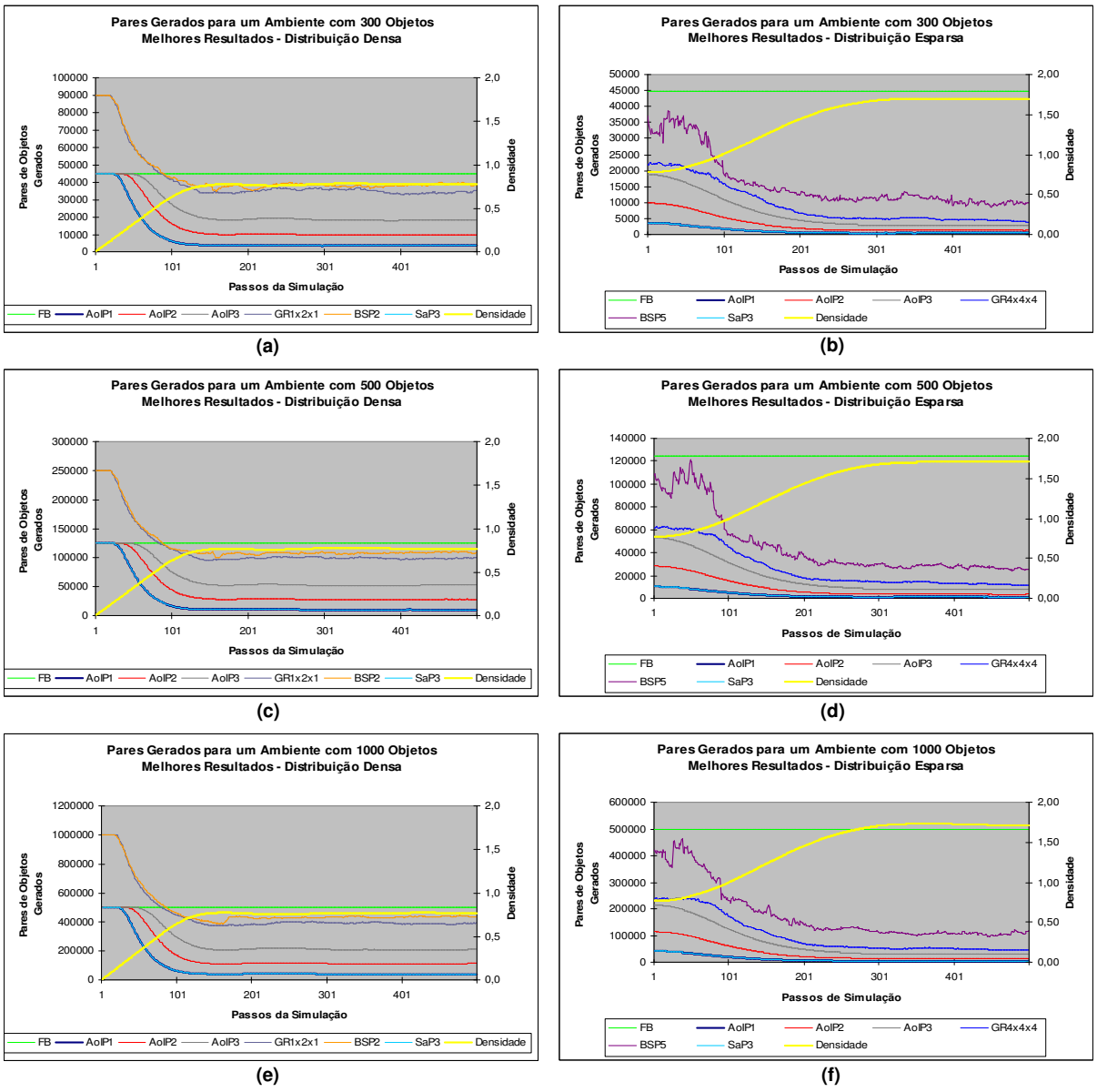


FIG. 9.8: Melhores resultados obtidos para uma distribuição densa e esparsa de 300, 500 e 1000 objetos.

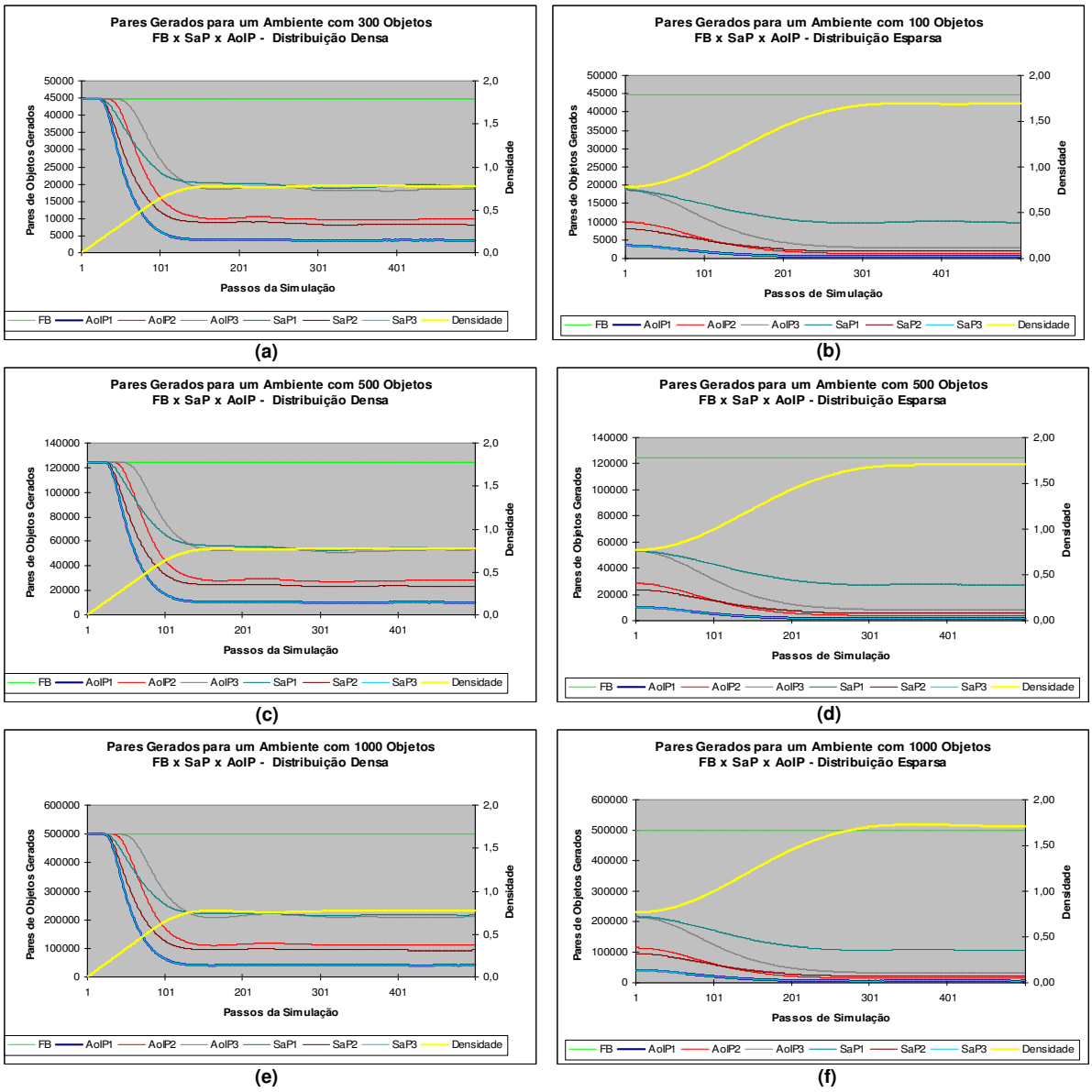


FIG. 9.9: Comparação entre os resultados obtidos para os métodos *AoIP* e *Sweep and Prune* para uma distribuição densa e esparsa de 300, 500 e 1000 objetos.

9.3 APÊNDICE 3: RESULTADOS PARA O TEMPO DE PROCESSAMENTO

Este apêndice apresenta graficamente os resultados do tempo médio de processamento para cada etapa da fase de descarte. Apresenta também, os valores médios totais do tempo de processamento para a Fase de Descarte em conjunto com o número de pares gerados nos cenários simulados de acordo com a quantidade de objetos considerada.

9.3.1 TEMPOS MÉDIOS PARA AS ETAPAS DA FASE DE DESCARTE

Nos gráficos das Figuras 9.10 a 9.19 são apresentados os tempos de processamento das etapas da fase de descarte para cada tipo de distribuição espacial adotada: em (a) valores obtidos para 100 objetos; em e (b) valores obtidos para 300 objetos, em (c) valores obtidos para 500 objetos e em (d) valores obtidos para 1000 objetos.

Os tempos de processamento para cada método foram medidos de acordo com as etapas de criação, atualização e geração de pares. Para cada método, as configurações adotadas em cada simulação foram ordenadas do menor ao maior tempo de processamento obtido.

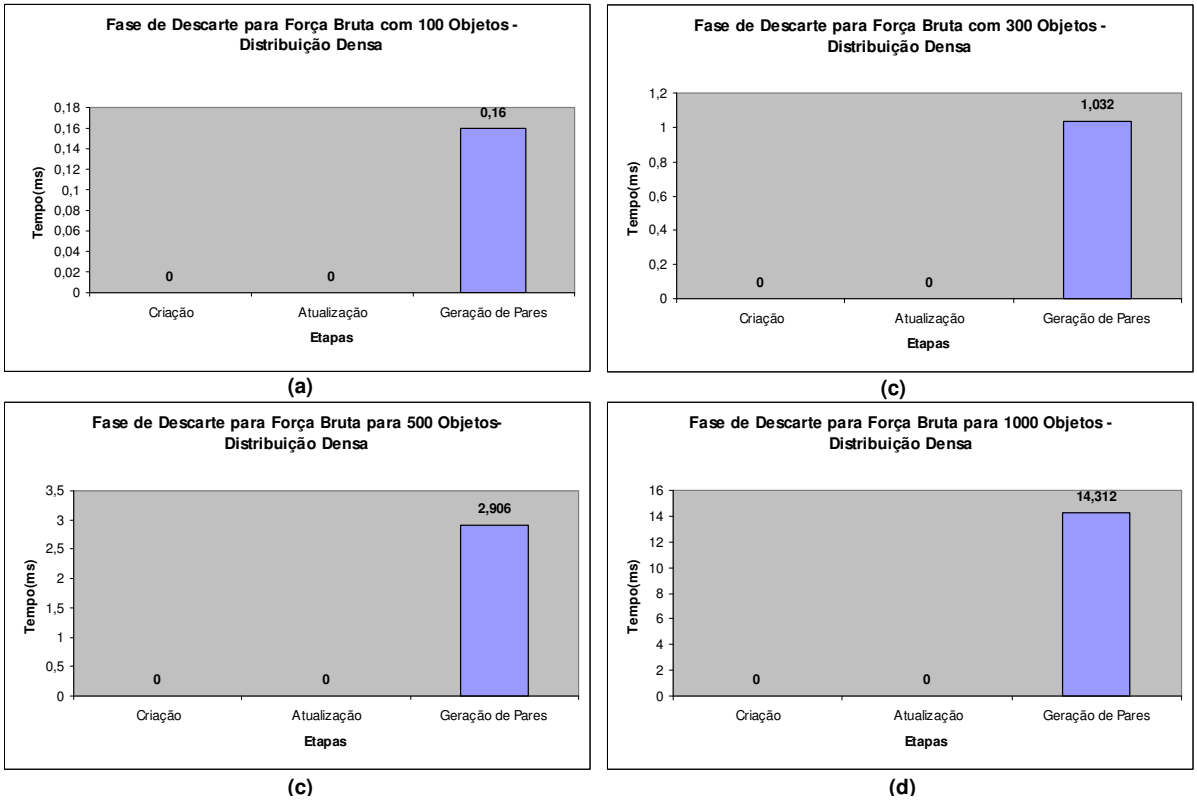


FIG. 9.10: Tempo de processamento da fase de descarte para o método *Força Bruta* para uma distribuição densa com 100, 300, 500 e 1000 objetos.

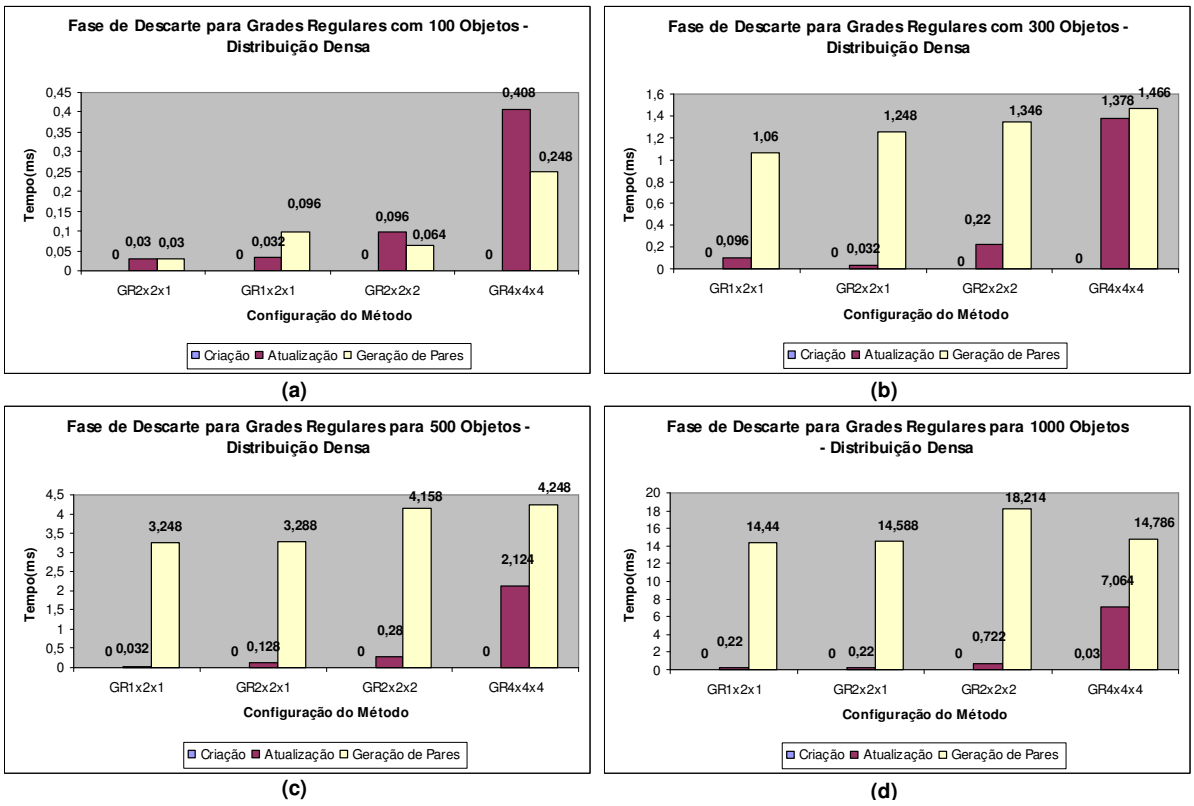


FIG. 9.11: Tempo de processamento da fase de descarte para o método *Grades Regulares* para uma distribuição densa com 100, 300, 500 e 1000 objetos.

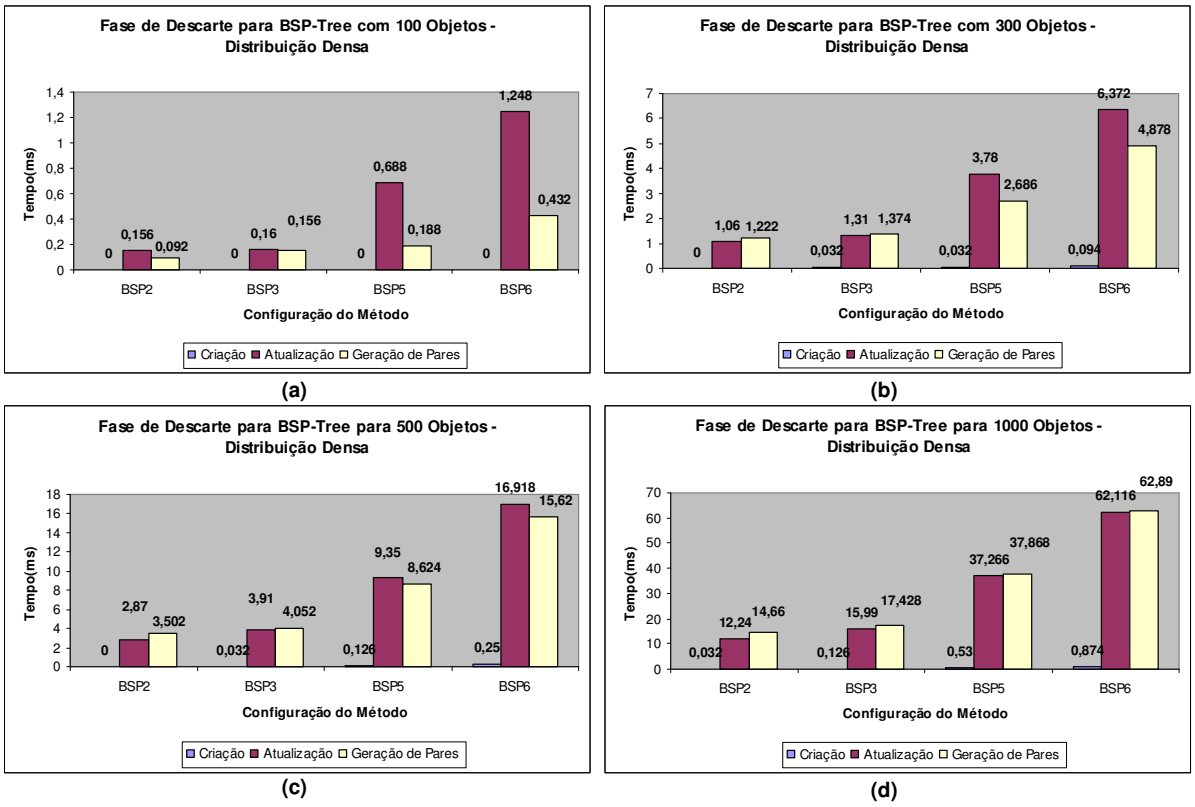


FIG. 9.12: Tempo de processamento da fase de descarte para o método *BSP-Tree* para uma distribuição densa com 100, 300, 500 e 1000 objetos.

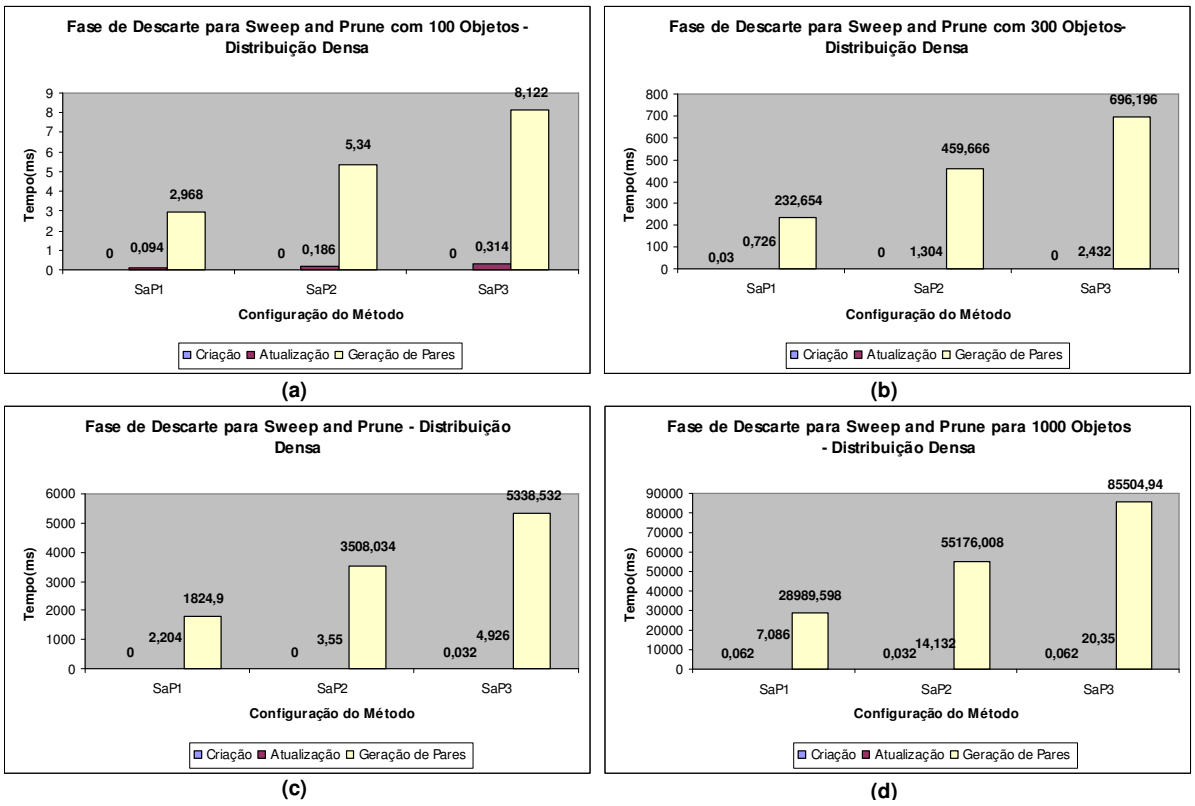


FIG. 9.13: Tempo de processamento da fase de descarte para o método *Sweep and Prune* para uma distribuição densa com 100, 300, 500 e 1000 objetos.

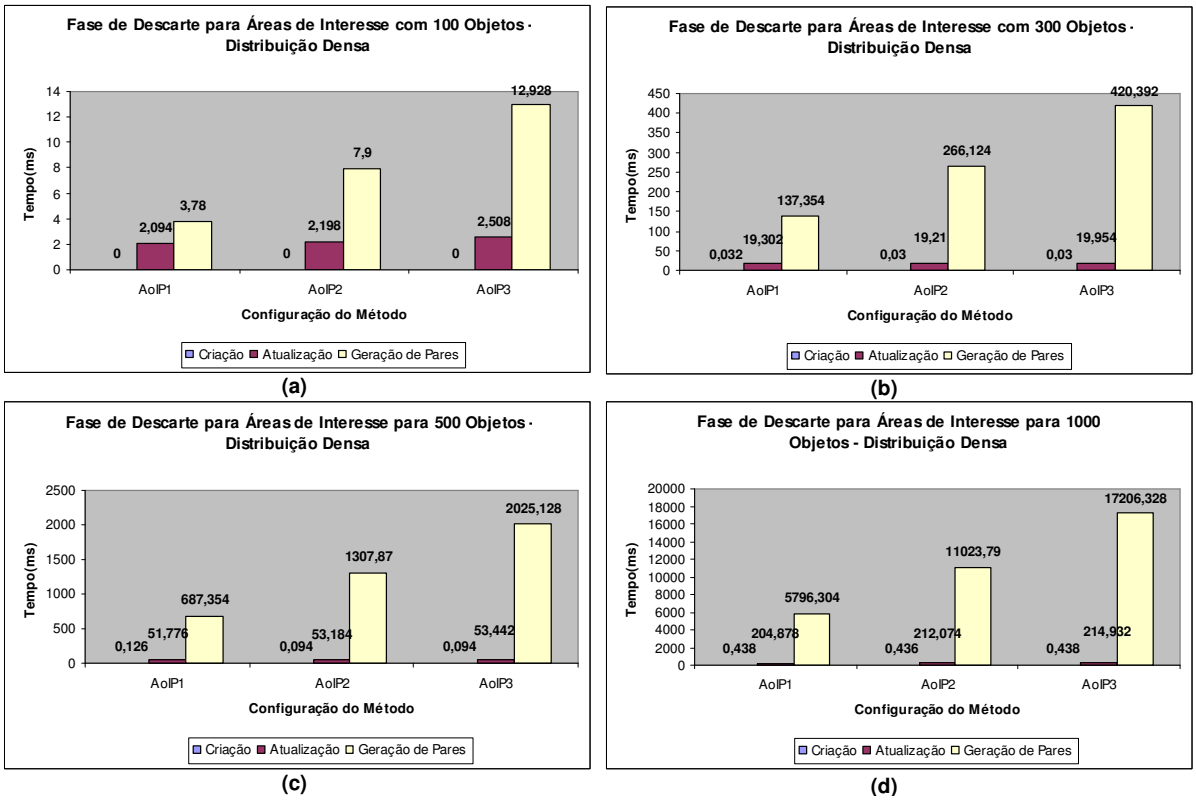


FIG. 9.14: Tempo de processamento da fase de descarte para o método *AoIP* para uma distribuição densa de 100, 300, 500 e 1000 objetos.

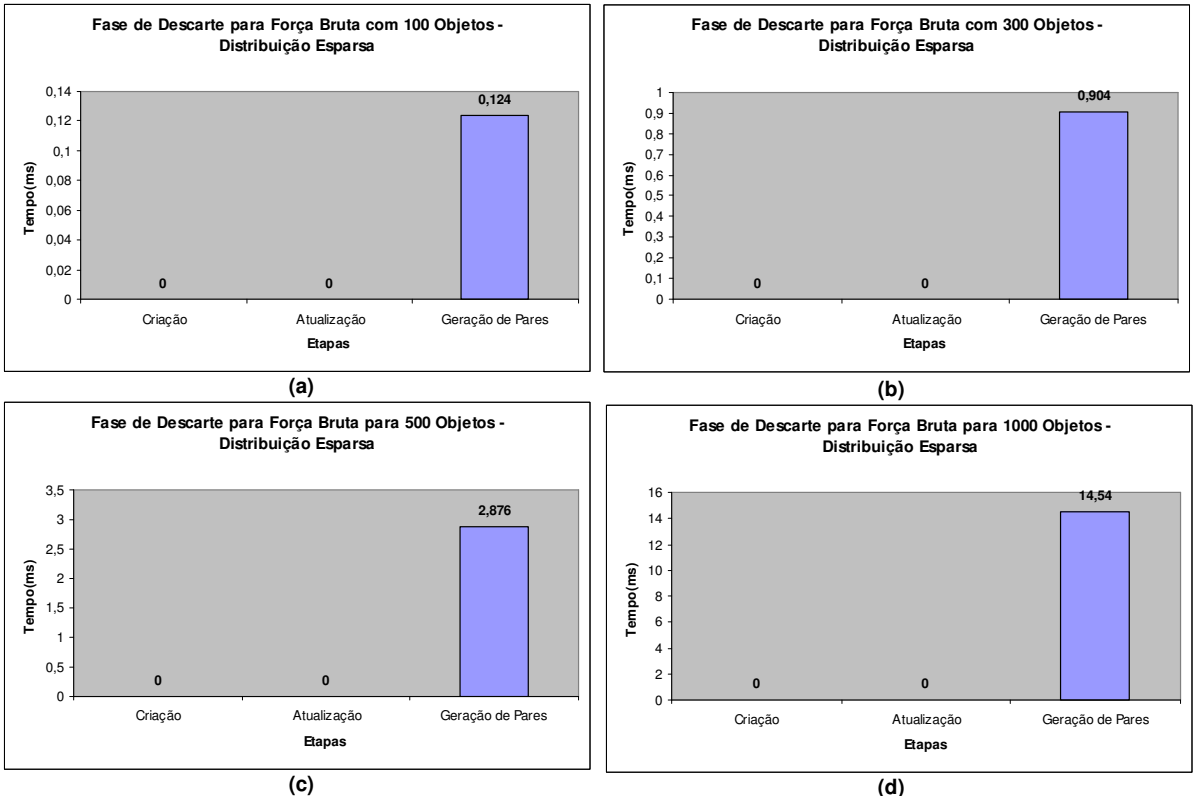


FIG. 9.15: Tempo de processamento da fase de descarte para o método *Força Bruta* para uma distribuição esparsa com 100, 300, 500 e 1000 objetos.

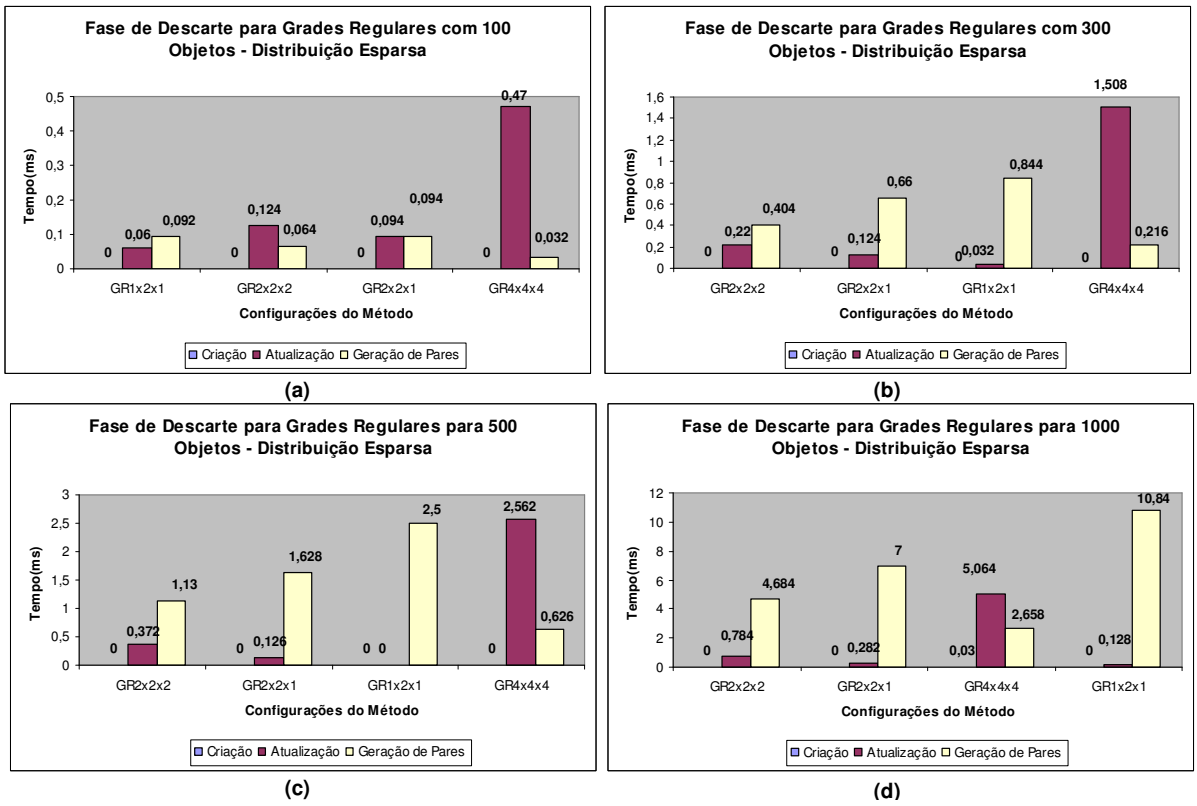


FIG. 9.16: Tempo de processamento da fase de descarte para o método *Grades Regulares* para uma distribuição esparsa com 100, 300, 500 e 1000 objetos.

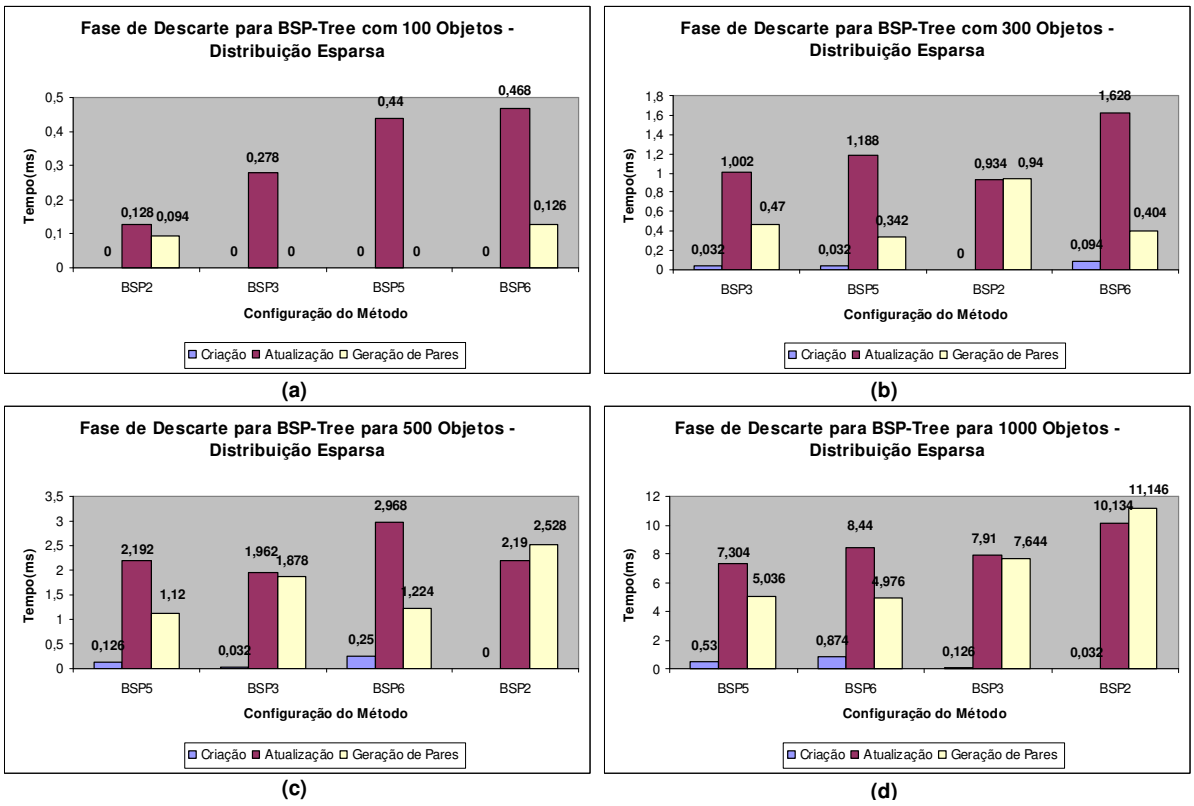


FIG. 9.17: Tempo de processamento da fase de descarte para o método *BSP-Tree* para uma distribuição esparsa com 100, 300, 500 e 1000 objetos.

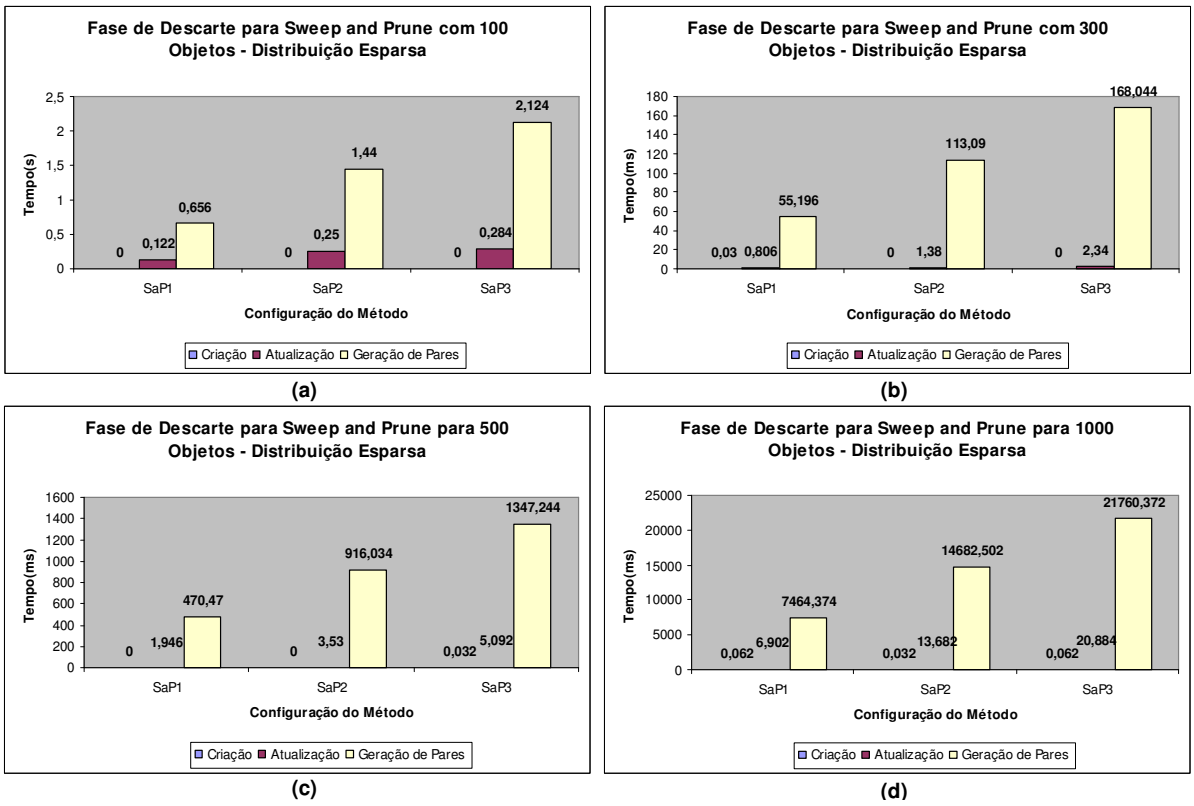


FIG. 9.18: Tempo de processamento da fase de descarte para o método *Sweep and Prune* para uma distribuição esparsa com 100, 300, 500 e 1000 objetos.

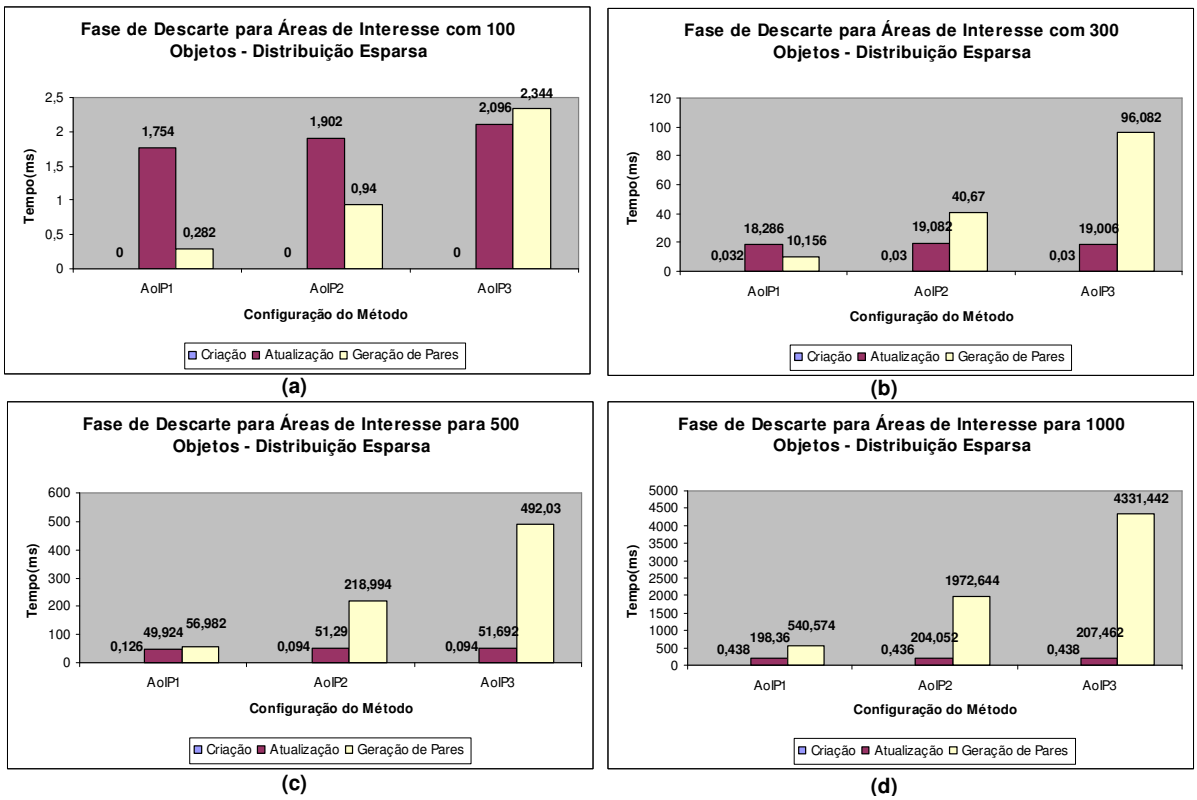


FIG. 9.19: Tempo de processamento da fase de descarte para o método *AoIP* para uma distribuição esparsa com 100, 300, 500 e 1000 objetos.

9.3.2 TEMPOS MÉDIOS DE PROCESSAMENTO DOS MÉTODOS DE DESCARTE

As Tabelas 9.1, 9.2, 9.3 e 9.4 apresentam os valores dos tempos médios totais de processamento obtidos nas simulações realizadas com 100, 300, 500 e 1000 objetos distribuídos densamente no ambiente. As Tabelas 9.5, 9.6, 9.7 e 9.8 apresentam os valores dos tempos médios totais obtidas nas simulações realizadas com 100, 300, 500 e 1000 objetos distribuídos esparsamente no ambiente. Os métodos encontram-se ordenados de acordo com o tempo médio de processamento.

TAB. 9.1: Tempo médio de processamento dos métodos de descarte para 100 objetos - distribuição densa

	Pares de Objetos Gerados	**Tempos de Processamento(ms)**
GR2x2x1	4.522,4	0,06
GR1x2x1	4.399,4	0,13
FB	4950	0,16
GR2x2x2	5.183,9	0,16
BSP2	4.760,3	0,24
BSP3	5.390,2	0,32
GR4x4x4	5.183,9	0,66
BSP5	10.960,4	0,87
BSP6	18.481,2	1,68
SaP1	2.565,6	3,06
SaP2	1.407,3	5,53
AoIP1	913,2	5,87
SaP3	913,2	8,44
AoIP2	1.663,8	10,09
AoIP3	2.557,8	15,44

TAB. 9.2: Tempo médio de processamento dos métodos de descarte para 300 objetos - distribuição densa

	Pares de Objetos Gerados	**Tempos de Processamento(ms)**
FB	44850	1,032
GR1x2x1	41.080,6	1,156
GR2x2x1	41.252,1	1,28
GR2x2x2	48.042,9	1,566
BSP2	43.684,1	2,282
BSP3	49.388,2	2,716
GR4x4x4	48.042,9	2,844
BSP5	100.649,9	6,5
BSP6	169.659,2	11,344
AoIP1	8.471,8	156,688
SaP1	23.165,9	233,41
AoIP2	15.317,9	285,364
AoIP3	23.367,3	440,376
SaP2	13.078,5	460,97
SaP3	8.471,9	698,628

TAB. 9.3: Tempo médio de processamento dos métodos de descarte para 500 objetos - distribuição densa

	Pares de Objetos Gerados	**Tempos de Processamento(ms)**
FB	124.750	2,906
GR1x2x1	115.885,5	3,28
GR2x2x1	117.082,6	3,416
GR2x2x2	135.265,6	4,438
GR4x4x4	135.265,6	6,372
BSP2	122.936,1	6,374
BSP3	139.162,8	7,996
BSP5	282.900,2	18,1
BSP6	476.774,7	32,788
AoIP1	23.589,3	739,256
AoIP2	43.019,2	1361,148
SaP1	64.448,3	1827,104
AoIP3	65.815,1	2078,664
SaP2	36.497,8	3511,584
SaP3	23.589,3	5343,49

TAB. 9.4: Tempo médio de processamento dos métodos de descarte para 1000 objetos - distribuição densa

	Pares de Objetos Gerados	**Tempos de Processamento(ms)**
FB	499.500	14,312
GR1x2x1	456.788,6	14,66
GR2x2x1	460.001,9	14,808
GR2x2x2	534.208,9	18,936
GR4x4x4	534.208,9	21,88
BSP2	488.911,2	26,932
BSP3	555.415,1	33,544
BSP5	1.122.297,2	75,664
BSP6	1.883.039,1	125,88
AoIP1	9.3291,1	6001,62
AoIP2	171.403	11236,3
AoIP3	262.980,9	17421,698
SaP1	256.763,8	28996,746
SaP2	144.422,3	55190,172
SaP3	93.291,1	85525,352

TAB. 9.5: Tempo médio de processamento dos métodos de descarte para 100 objetos - distribuição esparsa

	Pares de Objetos Gerados	**Tempos de Processamento(ms)**
FB	4950	0,124
GR1x2x1	3.433,5	0,152
GR2x2x2	1.649,9	0,188
GR2x2x1	2.332,5	0,188
BSP2	3.368,9	0,222
BSP3	2.478,2	0,278
BSP5	1.677,1	0,44
GR4x4x4	967,7	0,502
BSP6	1.543,2	0,594
SaP1	1.277,1	0,778
SaP2	350,1	1,69
AoIP1	111,2	2,036
SaP3	111,2	2,408
AoIP2	322,3	2,842
AoIP3	650,3	4,44

TAB. 9.6: Tempo médio de processamento dos métodos de descarte para 300 objetos - distribuição esparsa

	Pares de Objetos Gerados	**Tempos de Processamento(ms)**
GR2x2x2	14.754,1	0,624
GR2x2x1	21.392,3	0,784
GR1x2x1	30.911,6	0,876
FB	44850	0,904
BSP3	24.147,5	1,504
BSP5	15.861,2	1,562
GR4x4x4	9219,1	1,724
BSP2	31.692,9	1,874
BSP6	14.574,6	2,126
AoIP1	1.061,7	28,474
SaP1	11.899,3	56,032
AoIP2	3.148,4	59,782
SaP2	3.337,5	114,47
AoIP3	6.413,2	115,118
SaP3	1.061,7	170,384

TAB. 9.7: Tempo médio de processamento dos métodos de descarte para 500 objetos - distribuição esparsa

	Pares de Objetos Gerados	**Tempos de Processamento(ms)**
GR2x2x2	41.387,1	1,502
GR2x2x1	60.726,1	1,754
GR1x2x1	87.380,3	2,5
FB	124750	2,876
GR4x4x4	25.922,1	3,188
BSP5	45.307,1	3,438
BSP3	66.318,4	3,872
BSP6	42.647,2	4,442
BSP2	89.742,9	4,718
AoIP1	3.021,9	107,032
AoIP2	8.951,1	270,378
SaP1	33.697,8	472,416
AoIP3	18.146,3	543,816
SaP2	9.719,1	919,564
SaP3	3.021,9	1352,368

TAB. 9.8: Tempo médio de processamento dos métodos de descarte para 1000 objetos - distribuição esparsa

	Pares de Objetos Gerados	**Tempos de Processamento(ms)**
GR2x2x2	159.025,7	5,468
GR2x2x1	233.714,4	7,282
GR4x4x4	99.695,1	7,752
GR1x2x1	342.442,6	10,968
BSP5	179.583,5	12,87
BSP6	168.478,9	14,29
FB	499500	14,54
BSP3	261.670,3	15,68
BSP2	355.898,3	21,312
AoIP1	11.930,6	739,372
AoIP2	35.453,9	2177,132
AoIP3	71.819,9	4539,342
SaP1	133.277,4	7471,338
SaP2	38.208,9	14696,216
SaP3	11.930,6	21781,318

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo