ESTUDO DA FORMAÇÃO DE RACHADURAS EM MATERIAIS SÓLIDOS PARA A PRODUÇÃO DE BLOCOS

Alexandre Moreira Dias[1], Fátima de Paula[1], Glauber Luis de Moraes[1], Valmir Aureliano da Silva[1], Marcio Magini[1]

[1]Universidade do Vale do Paraíba, Estrada do Limoeiro 250 - Jardim Dora
CEP 12305-810 - Jacareí – SP,
valmir_programador@yahoo.com.br

Resumo - Este artigo apresenta uma aplicação de um problema de otimização e busca, com o auxilio de algoritmos computacionais orientados a objetos, aplicada a um estudo sobre a ocorrência de rachaduras, a partir da formação de bolhas em sistemas sólidos. Para realização deste estudo, desenvolveu-se um sistema computacional, utilizando as variáveis discretas: altura, largura e profundidade que representam uma matriz tridimensional na qual verifica-se o caminho de maior custo para percorrer toda a matriz. O sistema foi submetido a uma bateria de testes para aferir a probabilidade de rachaduras em sólidos de acordo com a quantidade de bolhas formadas. Os resultados obtidos servirão para analisar a resistência de determinados materiais sólidos.

Palavras-chave: Resistência de Materiais, Otimização e Problemas Computacionais.

Área do Conhecimento: Ciência da Computação, Engenharia da Computação, Resistência dos Materiais e Engenharia Civil.

Introdução

Muitos problemas computacionais são modelados partindo-se de um objetivo e de um conjunto de hipóteses ou restrições e estas balizam a solução dos problemas. Elas vão desde restrições pára a produção de materiais até a maneira mais eficiente para a alocação de máquinas. Esses tipos de problemas são conhecidos como JobShop **[1]**.

As técnicas computacionais mais comuns aplicadas a estes modelos são: equações diferenciais, programação linear, programa não linear e métodos estatísticos (Monte Carlo, Fuzzy, Booleana, entre outros) **[2]**. Este trabalho irá basear-se em um dos métodos estatisticos somente,

O método de Monte Carlo é uma das ferramentas mais importantes e mais utilizadas em finanças, que permite: calcular valores de instrumentos financeiros mesmo que não seja conhecida parametricamente a distribuição de probabilidades subjacente e resolver uma quantidade extensa de problemas através da simulação de cenários e o posterior cálculo de um valor esperado. Importante na implementação deste método é o uso de um gerador de números aleatórios confiável e de um algoritmo de redução de variância.

A lógica *fuzzy* consiste em aproximar a decisão computacional da decisão humana, isto é, fazer com que uma máquina não apresente apenas soluções binárias como "sim" ou um "não", essa lógica gera a possibilidade de decisões "abstratas", do tipo "mais ou menos", "talvez sim", e outras tantas variáveis que representem as decisões humanas. O primeiro passo para desenvolver um sistema deste tipo é determinar as variáveis a serem utilizadas como dados de entrada, a partir dos quais o sistema gerará o mapa.

A álgebra booleana é um sistema de dedução matemática restrito aos valores zero e um (falso e verdadeiro). Operadores binários definidos para este conjunto de valores aceitam um par de entradas booleanas e produzem um valor booleano único. Por exemplo, o operador booleano AND aceita duas entradas booleanas e produz uma única saída booleana (o AND lógico das duas entradas).

O objetivo desse trabalho é mostrar de forma simplificada, através de programação orientada a objetos e lógica booleana, como resolver um problema de otimização e busca. Mais especificamente quer-se determinar através de um algoritmo desenvolvido em linguagem Java o caminho mais provável que interpola um conjunto de pontos sujeitos às restrições impostas pela simulação de diversas situações físicas e aplicações de engenharia traçando um paralelo entre os resultados gerados e o processo de estudo de falhas em sólidos.

Materiais e Métodos

Para o desenvolvimento do sistema de simulação utilizou-se do kit J2SDK 1.4.0.7 (Java 2 Standard Development Kit)**[3]** e do editor de códigos Jcreator LE 2.5**[4]** e para o processamento do algoritmo um computador Intel Celeron, 512 Mbytes de RAM, Clock 2,6GHz. Windows XP SP2 ®.

A implementação do sistema seguiu os seguintes passos:

1) Foi elaborada uma matriz tridimensional no formato **9 x 9 x 9** possuindo valores que representam as bolhas formadas no sólido, sendo que quanto menor a bolha maior é a probabilidade da mesma ocorrer no sólido Estas bolhas podem variar de tamanho e quantidade. A tabela 01 apresenta como a probabilidade de aparecimento de bolhas foi distribuída através das camadas na matriz.

Tabela 1: Probabilidade da formação de bolhas por camada.

Tamanho das Bolhas	Porcentagem por Camada
1	15%
2	14%
3	13%
4	11%
5	9%
6	7%
7	6%
8	4%
9	2%

2) Este passo trata do processamento dos dados gerados na matriz, ou seja, verifica-se nesta etapa qual o caminho que possui o maior custo sendo este associado com a maior probabilidade de rachaduras. A ordem de busca pelo caminho de maior custo segue os seguintes critérios: **a–** Definir as regras de oscilação entre as camadas, ou seja, especificar quantas vezes o sistema poderá retornar para a camada de nível anterior. **b–**Selecionar a primeira célula da primeira camada e descobrir o vizinho de maior custo da referida célula. Este vizinho pode estar na camada superior ou na camada atual ou, ainda, na camada inferior, dependendo das configurações de inicialização. **c–**Ir para a próxima célula encontrada com maior custo e marcar célula anterior como visitada. **d–** Os passos de b a c repetir-se-ão até o sistema atingir a camada de número 8. **e–** O sistema voltará ao passo b e testará a célula número dois da primeira camada até que todas as células da primeira camada tenham sido processadas. **f–**Compara todos os resultados encontrados e seleciona o que contenha o maior custo entre os caminhos encontrados. A figura 1 ilustra o processamento de uma célula em busca de seu vizinho de maior custo.

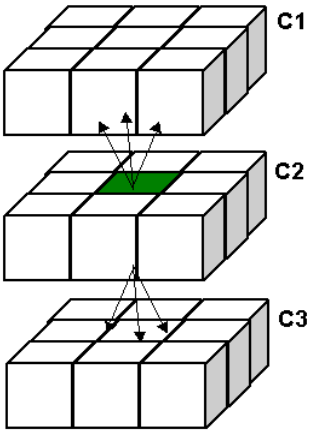

Figura 1. – Célula da camada C2 em busca do maior vizinho que pode estar nas 9 posições da camada superior C1 ou nas 8 posições da própria camada C2 ou ainda nas 9 posições da camada inferior C3.

3) Este passo trata da plotagem (gráfica ou linha de comandos) dos resultados obtidos no processamento (passo 2).

Sendo assim, foram criadas 3 classes principais e uma auxiliar. *Classe Matriz* que contém todos os métodos necessários para manipular os valores e atributos do sólido, *Classe Processamento:* efetua os cálculos necessários para encontrar o caminho de maior custo, Classe *DrawShapes* plota os dados utilizando uma interface gráfica e a classe P*rincipal* utilizada para exibir os resultados em ambiente não gráfico.

O Sistema e suas Variáveis

Variáveis de Visão (As rachaduras nos diversos sólidos são visualizadas de três ângulos diferentes)

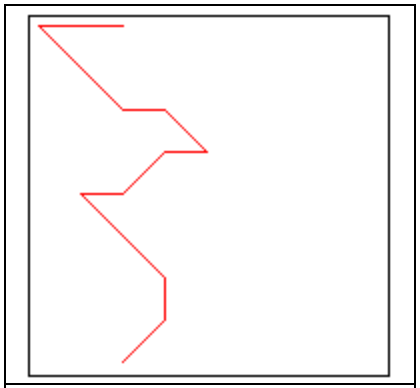

Figura2: Visão Frontal do sólido

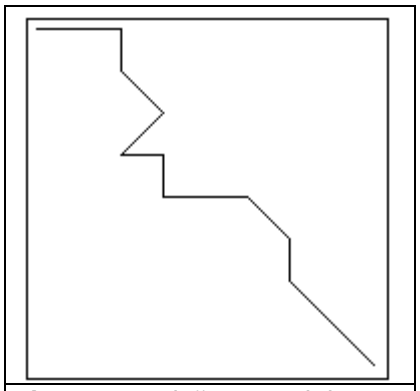

Figura 3. – Visão Lateral do sólido.

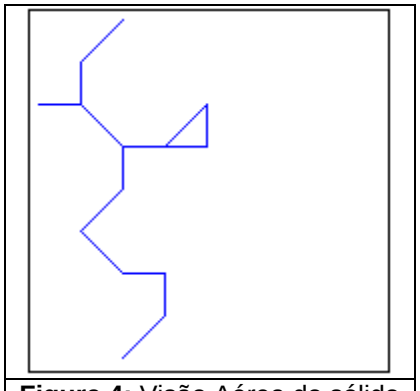

Figura 4: Visão Aérea do sólido

Camadas sobrepostas (são nove camadas no total)

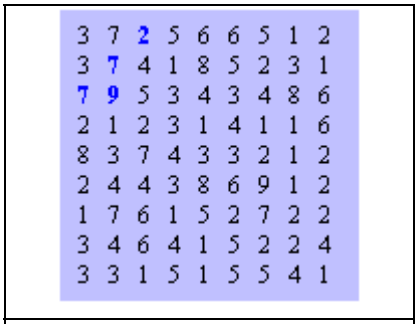

Figura 5. – Visão computacional da primeira camada do sólido. Os pontos azuis são as posições percorridas na camada.

Totais e Probabilidades.

- Total de Células Percorridas: Posicionado no rodapé do formulário indica o total de células que foram processadas para a simulação.
- Soma das Células Percorridas: Indica a soma de todas as células percorridas durante a simulação.
- Probabilidade de Quebra: Indica a probabilidade de rachadura em uma determinada peça, utilizando a seguinte formula:

$$Pq = \frac{SCP * 100}{TCP * 9}$$

Pq = Probabilidade de Quebra.
SCP = Soma das Células Percorridas
TCP = Total de Células Percorridas.

- Tempo de Processamento: Indica o tempo gasto pelo processador para simular a probabilidade de rachadura.

Resultados e Discussão

A fase de processamento da matriz executada pela classe Processamento é alicerçada em três métodos: *superiorCamada* que efetua a busca do maior vizinho da camada superior, *midiCamada* que efetua a busca do maior vizinho da camada atual e *inferiorCamada* que efetua a busca do maior da camada inferior, sendo que ao atingir a última camada o resultado encontrado é armazenado para posterior comparação do caminho de maior custo.
Uma bateria de testes com configurações de caminhamentos diferentes (que simbolizam os diferentes sólidos) e com plataformas/arquiteturas diferentes foram executados para o complemento

deste artigo. Estas configurações permitem estabelecer a quantidade de vezes que uma camada superior pode ser percorrida novamente. A Tabela 2 apresenta os testes efetuados com configurações de oscilações diferentes, ou seja, a quantidade de vezes que será permitido uma célula retornar ao nível de profundidade anterior. A tabela 3 apresenta testes efetuados com plataformas/arquiteturas diferentes verificando o desempenho do sistema nas diferentes plataformas computacionais e as figuras 2,3, 4 e 5 demonstram as variáveis do sistema.

Tabela 2. – Testes efetuados com configurações de oscilações diferentes.

TP: Tempo de Processamento

PQ: Probabilidade de Quebra do sólido mediante a formação de bolhas.

CP: Células Percorridas

SCP: Soma das Células Percorridas

TS: Tipo de Sólido(É definido pela quantidade de oscilações)

TP.	PQ.	CP.	SCP	TS.
0,060	79,63%	18	129	Tipo 01
0,040	67,68%	22	134	Tipo 02
0,050	78,33%	20	141	Tipo 03

Tabela 3. – Testes efetuados com plataformas/ arquiteturas diferentes

TP: Tempo de Processamento

PQ: Probabilidade de quebra do sólido mediante a formação de bolhas.

CP: Células Percorridas

Config: Configuração da Plataforma Operacional

TP.	PQ.	CP.	Config.
0.000	72,92%	16	Intel Celeron, 512 MB RAM,2,6GHz. Windows XP-Prof.
0,005	77,52%	16	Intel Celeron, 512 MB RAM,2,6GHz.Linux Ubuntu 5.10.
0,020	73,08%	18	MAC MINI Power-PC G4, 256M RAM, 1.4GHZ, MAC OS X
0,181	80,00%	17	Intel Celeron, 128 MB RAM, clock 400 MHz. Windows 98.

Estes testes foram realizados para verificar a estabilidade e robustez das classes e métodos criados. Os resultados, apresentados nas tabelas anteriores, foram considerados satisfatórios.

Conclusão

Com o auxilio do sistema de simulação é possível verificar como ocorre o processo de formação de rachaduras em sólidos. Assim a utilização dos códigos computacionais desenvolvidos neste trabalho que visam otimizar a busca de bolhas formadas em sólidos associada com o conhecimento prévio de características técnicas dos materiais de construção permitem que o usuário crie uma estratégia que melhor se adapte aos seus interesses, possibilitando direcionar melhor os recursos para utilização do mesmo e, consequentemente, poderá obter maior segurança e retorno de seus investimentos.

Assim a utilização dos códigos de otimização e busca desenvolvidos, foi possível mapear possíveis falhas no processo de formação e produção de sólidos.

Referências

[1]- http://www.dep.fem.unicamp.br/unisi m/ publicacoes/tese_lincoln.pdf acessado em 23/03/2006

[2]-http://www.if.ufrj.br/teaching/sergio/ demo/ demon.html acessado em 10/05/2006

[3] http://www.sun.com acesso em 10/05/2006

[4] http://www.jcreator.com acesso em 10/05/2006

[5]-http://sensei.ieec.uned.es/cgi-bin/aepia/contenidoNum.pl?numero=19 acessado em 23/05/2006

[6]-http://www.pr.gov.br/batebyte/edicoes/1994/bb31/simulacao.htm acessado em 23/05/2006

[7]http://jcoelho.m6.net/edicao3.asp?pa=2078 acessado em 24/05/2006

[8] DEITEL, H.M; DEITEL, P.J Java**:** como programar. 4. ed. Porto Alegre: Bookman, 2003. 1386p.