E P 1
Algoritmos de Busca

Inteligência Artificial
MAC 5739 - MAC 415

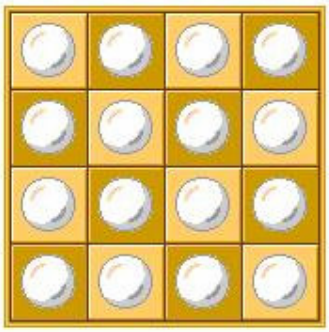
Leliane Nunes de Barros
2007

1 Objetivo

Implementar os algoritmos de busca estudados na disciplina de Inteligência Artificial.

2 O jogo: FIVER

Dado um tabuleiro NxN composto por N*N peças brancas, o objetivo desse jogo é converter todas as peças brancas do tabuleiro em peças pretas. Selecionar uma peça P tem o efeito de trocá-la de cor, bem como trocar a cor de todas as peças adjacentes a P (exceto as peças das diagonais). A solução ótima é aquela com o menor número de seleções.



No site http://thinks.com/java/fiver/fiver.htm você encontra uma versão online do jogo. Para selecionar uma peça basta *clicá-la* com o mouse. A solução ótima é aquela com o menor número de cliques. Para compreender melhor o jogo FIVER, tente descobrir as soluções ótimas para os diferentes tamanhos de tabuleiro.

Seja clicar(i,j) a ação de selecionar a posição (i,j) do tabuleiro. Um exemplo de solução ótima para o Fiver 3x3 é dada pela seguinte seqüência de passos:

clicar(1,1) ➔ clicar(3,3) ➔ clicar(2,2) ➔ clicar(3,1) ➔ clicar(1,3)

3 O que você deve implementar

Implementar um programa capaz de resolver instâncias NxN do jogo FIVER, usando as seguintes estratégias de busca:

- Busca em Largura (*Breadth First Search - BrFS*)
- Busca em Profundidade (*Depth First Search - DFS*)
- Busca em Profundidade Iterativa (*Iterative Deepening Search - IDS*) (com passo=1)
- Busca A*
- Busca IDA*

Seu programa deve considerar como entrada o tamanho do tabuleiro (N) e um parâmetro que especifique o tipo de busca (*BrFS*, *DFS, IDS, A** e *IDA**). Seu programa deve imprimir:

- o caminho da solução, dada pela seqüência de ações clicar(i,j);
- a profundidade do estado meta encontrado (ou seja, o tamanho da solução);
- o número total de nós gerados e o número de nós visitados (nós gerados que já foram testados e para os quais, se possível, foram gerados seus sucessores);
- o valor médio do fator de ramificação da árvore de busca (*branching factor*) (cálculado como a razão entre o número total de nós sucessores gerados e o número de nós visitados).

4 Como evitar o processamento redundante na busca

Durante a busca, um mesmo estado pode ser visitado muitas vezes. Para melhorar a eficiência da busca é preciso evitar o processamento de nós repetidos. Se um nó já foi visitado então ele não deve ser incluido na lista de nós a serem visitados. Você deve usar uma tabela de *hashing* para manter registro dos estados já visitados. Se o seu EP não evitar nós repetidos sua nota será ZERO.

5 Função sucessor de estado

Para facilitar a correção dos EPs, a geração dos nós sucessores deve obedecer uma ordem fixa, selecionando-se as peças a partir da posição (1,1) até a posição (N,N), seguindo as linhas horizontais do tabuleiro. Essa ordem de geração dos sucessores de um nó, que chamaremos de *ordem dos sucessores,* afeta a construção da fila de nós a serem visitados da seguinte forma:

1. nas buscas BrFS, DFS e IDS, a inserção de nós sucessores na lista de nós a serem visitados deve obedecer a *ordem dos sucessores*;
2. no caso da lista de prioridade para as buscas A* e IDA*, a inserção de nós na fila de prioridades deve obedecer a *ordem dos sucessores* para nós de mesmo

valor de avaliação de custo (isto é, f(n)). Para o caso de empates entre nós sucessores de pais diferentes, a inserção deve obedecer a ordem cronológica, ou seja, nós gerados mais recentemente devem ser ordenados antes do que os nós já pertencentes à fila.

6 Funções heurísticas para o jogo FIVER

Você pode construir mais do que uma função heurística para comparar os resultados. Para cada uma das funções heurísticas usadas, você deve provar que ela é admissível. Para cada função heurística adicional o aluno ganhará um bônus, a ser especificado, em sua nota geral.

7 Linguagem de programação

Você deverá fazer o seu EP na linguagem Java (no EP2 vocês usarão esses mesmos programas com uma biblioteca Java específica). A entrega será eletrônica através do sistema Moodle.

8 Arquivos que deverão ser entregues

1. Cada aluno deverá entregar um único arquivo fonte, chamado ***search***, contendo um cabeçalho especificando nome e número USP do aluno.
2. Um arquivo pdf (chamado **`relatorio.pdf)`** entitulado **ANÁLISE DOS MÉTODOS DE BUSCA**, onde você deverá fazer uma breve definição dos métodos de busca (incluindo alguns detalhes de suas implementações) e uma análise dos diferentes métodos de busca aplicados ao jogo FIVER. Nesta análise, você deverá mostrar, sempre que possível, as propriedades estudadas sobre as diferentes estratégias de busca e das diferentes heurísticas que você elaborou para o jogo. Por exemplo, você poderá comparar o número de nós expandidos, memória utilizada em cada uma das buscas, etc. Essa análise deve ser feita construindo-se gráficos em que o eixo X represente o tamanho do problema (isto é, X = N = 1, 2, ..., MAX, sendo MAX o maior tamanho de tabuleiro que você conseguir resolver). No eixo y, coloque os valores de número de nós expandidos, memória utilizada, tempo de CPU, etc. Você poderá fazer um gráfico com várias curvas[1] para cada tipo de busca (com todas as informações relevantes no eixo y) e gráficos adicionais que evidenciem propriedades específicas para comparar as buscas entre si. Por exemplo, compare o uso de memória de todas as buscas implementadas para os problemas de N=1, 2, ..., MAX, num único gráfico.

A nota do EP1 será a média entre a nota do relatório (entre 0 e 10) e do programa (entre 0 e 10), sendo que ambas deverão ser maior ou igual a 5, caso contrário, a nota final será a menor delas.

---

[1] Como os tamanhos dos problemas para FIVER são definidos por valores discretos, as curvas em função do tamanho dos problemas não são contínuas. No entanto, podemos ligar os pontos do gráfico por segmentos de retas.

9 Instruções gerais

Faça o seu programa bem estruturado, bem documentado e **sozinho**. Tenha o cuidado de implementar os algoritmos de busca de maneira independente do domínio de aplicação, ou seja, de forma que eles possam ser usados em outros jogos ou problemas sem que você tenha que mexer nos algoritmos de busca, mas somente na descrição de estado, funções geradoras de estados e teste de estado meta.